やさしいJavaScriptのきほん

Mana Takahashi
高橋麻奈 著

SB Creative

やさしい JavaScript のきほん

Mana Takahashi
高橋麻奈 著

まえがき

今日、Webにかかわる機会は多くなっています。Webを制作するためにはさまざまな技術が使われます。Webページを作成する「HTML」や、デザインを行う「CSS」など、たくさんの技術があります。学校や仕事でこれらの技術を学び、一通りのWebページが制作できるようになった！という方も多いかもしれません。

けれども、Webページをプログラミングするための「JavaScript」はどうでしょうか？　Webページを一通り制作できるようになったけれども、プログラミングはちょっと敷居が高いかなぁ・・・と思われている方は多いかもしれません。JavaScriptを使ったWeb開発を学んでいきたいと思っているけれども、なかなか手が出せないなぁ・・・と感じていらっしゃる方は多いことでしょう。

本書はJavaScriptの知識をわかりやすく解説するように心がけました。プログラミングになじみのない方にとっても、無理なく学習できるように構成されています。本書にはたくさんのサンプルページが掲載されています。実際に自分でWebページを作成し、プログラムの動作を確認しながら読み進めることができます。学んだ知識をWeb制作・開発に生かしていくためのアイデアやヒントを得ることもできるでしょう。

JavaScriptの基本を学び、ぜひWeb制作・開発技術を高めていってみてください。

本書が読者のみなさまのお役にたつことを願っております。

著者

目次

第1章 準備体操しよう！ ………… 1

1-1 JavaScriptでやってみよう！ ………… 2

Webページを動かそう ………… 2
どうしてWebページが動くの？ ………… 2
Column | JavaScriptとWebの歴史 ………… 5

1-2 ツールを使ってみよう ………… 6

ツールを用意しよう ………… 6
JavaScriptを入力してみよう ………… 8
JavaScriptを動かしてみよう ………… 10

1-3 しくみはどうなってるの？ ………… 12

Webのしくみを知ろう ………… 12

1-4 ファイルを用意しよう ………… 15

必要なファイルをおぼえておこう ………… 15
Column | HTMLの要素 ………… 16
Column | JavaScriptの入力場所 ………… 17
Column | ファイルの場所を指定するには ………… 19

第2章 JavaScriptをはじめよう！ ………… 21

2-1 時刻を表示しよう ………… 22

チャレンジ! 時刻を調べよう ………… 22
時刻を表示するプログラム ………… 22
順番に処理される ………… 24
オブジェクトを利用する ………… 24
Column | オブジェクトの種類 ………… 28

2-2 オブジェクトを使いこなそう ………… 29

チャレンジ! オブジェクトを活用しよう ………… 29
オブジェクトの機能を利用するには？ ………… 29
応用! さまざまな日時を表示してみよう ………… 31

Column ｜コメント …… 33
テクニックを学ぼう …… 33
Column ｜Documentオブジェクト …… 35
2-3 状況に応じた処理をしよう …… 36
チャレンジ! 平日・休日を表示しよう …… 36
曜日によって違う画像を表示するには？ …… 36
場合分けをしよう …… 38
場合分けのバリエーション …… 39
応用! 日時によって違う処理をする …… 40
「条件」を考えよう …… 42
複雑な条件も考えよう …… 43
応用! オープンとクローズの表示分け …… 44
2-4 繰り返し処理をしよう …… 45
チャレンジ! 画像を全部表示しよう …… 45
繰り返して表示しよう …… 46
繰り返し処理を書いてみよう …… 47
Column ｜インクリメント・デクリメント …… 48
繰り返しのバリエーション …… 48
応用! 繰り返して処理しよう …… 49
Column ｜リストと表 …… 52
繰り返し文を使いこなそう …… 52
Column ｜繰り返しの中の繰り返し …… 53
2-5 きめ細かく処理しよう …… 54
チャレンジ! 組み合わせて処理しよう …… 54
偶数だけ表示しよう …… 54
計算をするには？ …… 55
応用! 処理を組み合わせていく …… 56
2-6 データを準備しよう …… 60
チャレンジ! データを準備してみよう …… 60
画像名データを用意する …… 60
データを準備するには？ …… 61
応用! 配列データを用意して活用する …… 62

2-7 実践！ カレンダーを作ろう …… 66

チャレンジ! オリジナルのカレンダーをつくろう …… 66

Step 1 概要を設計する …… 66

Step 2 コードを考える …… 67

Step 3 カレンダーを完成させる …… 68

第3章 JavaScriptで動かそう！ …… 71

3-1 クリックで表示しよう …… 72

チャレンジ! クリックで動作するようにしよう …… 72

ユーザーの操作で動作させるには …… 72

クリック時の動作を確認しよう …… 74

イベント時に動作させる …… 75

応用! イベントの処理をしよう …… 76

Column｜関数 …… 78

3-2 画像を入れ替えよう …… 79

チャレンジ! 画像を入れ替えるには？ …… 79

画像を入れ替えるプログラム …… 79

画像を入れ替えるためには？ …… 80

応用! 画像を順に表示しよう …… 81

3-3 ランダムに表示しよう …… 84

チャレンジ! サイコロの目を表示しよう …… 84

ランダムに画像を表示する …… 84

ランダムに表示するには？ …… 86

Column｜Mathオブジェクト …… 86

応用! ランダムな数を使いこなそう …… 87

3-4 ページを操作しよう …… 91

チャレンジ! ページの一部を入れ替える …… 91

ページの要素を操作する …… 91

DOM …… 92

オブジェクトの操作 …… 94
応用! DOMで操作する …… 96
Column | 引数 …… 101
3-5 タイマーを使おう! …… 102
チャレンジ! 自動的に処理しよう …… 102
タイマーを使うには? …… 103
タイマーで画像を入れ替える …… 104
3-6 スライドショーにしよう …… 106
チャレンジ! スライドショーにする …… 106
Step 1 画面全体を設計する …… 107
Step 2 サムネイル部分を設計する …… 107
Step 3 ボタンの動作を設計する …… 108
Step 4 タイマーを設計する …… 108
Step 5 スライドショーを完成させよう …… 109

第4章 JavaScriptでチェックしよう! 113
4-1 入力フォームを作ろう …… 114
チャレンジ! 入力できるようにしよう …… 114
フォーム部品の種類 …… 114
フォームを使うには? …… 116
入力されたデータを調べる …… 117
Column | フォームの送信 …… 119
4-2 入力をチェックしよう …… 120
チャレンジ! フォームのチェック機能を充実させる …… 120
入力をチェックする …… 120
フォームへの入力を確認するには? …… 123
応用! 入力チェックのバリエーション …… 123
パターンを指定してチェックする …… 128
Column | 正規表現 …… 128

4-3 選択商品を表示しよう …… 129

チャレンジ! 選んだ商品を表示しよう …… 129

商品を表に追加するには？ …… 129

選択商品を表示しよう …… 131

4-4 合計金額を計算しよう …… 133

チャレンジ! 商品価格を計算しよう …… 133

価格表を用意する …… 133

小計を計算する …… 134

選択をクリアする …… 135

計算プログラムを完成させる …… 135

Column｜データベース …… 138

4-5 端末に応じたサイトにしよう …… 139

チャレンジ! 端末別の表示にしてみよう …… 139

Step1 Webページの内容を検討する …… 140

Step2 レイアウトを設計する …… 141

Step3 2種類のレイアウトを作成する …… 142

Step4 端末別のレイアウトで表示する …… 143

Column｜Navigatorオブジェクト …… 146

Step5 高機能なプログラムを作成するためには …… 146

Column｜jQuery …… 147

第5章 グラフィックを描こう！ …… 149

5-1 キャンバスを使おう！ …… 150

チャレンジ! キャンバスに描画してみよう …… 150

キャンバスを使うには？ …… 151

サイコロを描画する …… 152

応用! キャンバスを使いこなす …… 154

Column｜描画のためのメンバ …… 159

5-2 アニメーションを作成しよう …… 160

チャレンジ! アニメーションをしよう …… 160

キャンバスとタイマーでアニメーション …… 160
アニメーションを作成する …… 161
応用！画像によるアニメーション …… 163

5-3 実践！ ペイントアプリを作ろう …… 165
チャレンジ！ペイントアプリを作成する …… 165
Step1 ペイントアプリの画面を設計する …… 166
Step2 マウスでペイントするには？ …… 166
Step3 マウスの処理を作成する …… 167
Column | Eventオブジェクト …… 169
Step4 ペイントアプリを完成させよう …… 169
Column | アプリの機能を増やしていく …… 171

第6章 マップを活用しよう！ …… 173

6-1 マップを利用しよう …… 174
チャレンジ！マップを表示しよう …… 174
Googleマップとは …… 175

6-2 マップを準備しよう …… 176
マップの準備方法 …… 176
マップを表示する …… 178
Column | マップの種類 …… 180

6-3 マーカーを表示してみよう …… 181
チャレンジ！マーカーを表示しよう …… 181
マーカーを表示するには？ …… 182
マーカーを表示しよう …… 183
Column | Ajax …… 185

6-4 緯度・経度情報を使おう …… 186
チャレンジ！住所情報を表示する …… 186
住所情報を検索するには？ …… 187
Column | ジオコーディング …… 188

6-5 マップを完成させる ･･･ 189
マップを完成させよう ･･･ 189

付録 ･･･ 193
Quick Reference ･･･ 194
リソース ･･･ 198
jQuery ･･･ 199
Google Maps APIキー ･･･ 201

索引 ･･･ 203

1-1 JavaScriptでやってみよう!

Webページを動かそう

私たちはこれからJavaScriptを使おうとしています。JavaScriptを使うと、一体どんなことができるのでしょうか？

JavaScriptを使えば、魅力的なWebサイトを開発することができます。撮影した数多くの写真を整理し、表示できます。カートに入れた商品をチェックし、合計金額を計算することもできます。マップを利用したWebサイトを作成することもできるでしょう。

JavaScriptはWebサイトを開発するときに使える、大変便利で応用範囲の広い技術なのです。

■Webサイトと JavaScript

どうしてWebページが動くの？

JavaScriptを使えばWebページを操作し、動かしていくことができるようになります。では、どうしてJavaScriptはWebページを動かすことができるのでしょうか？

それは、JavaScriptが、

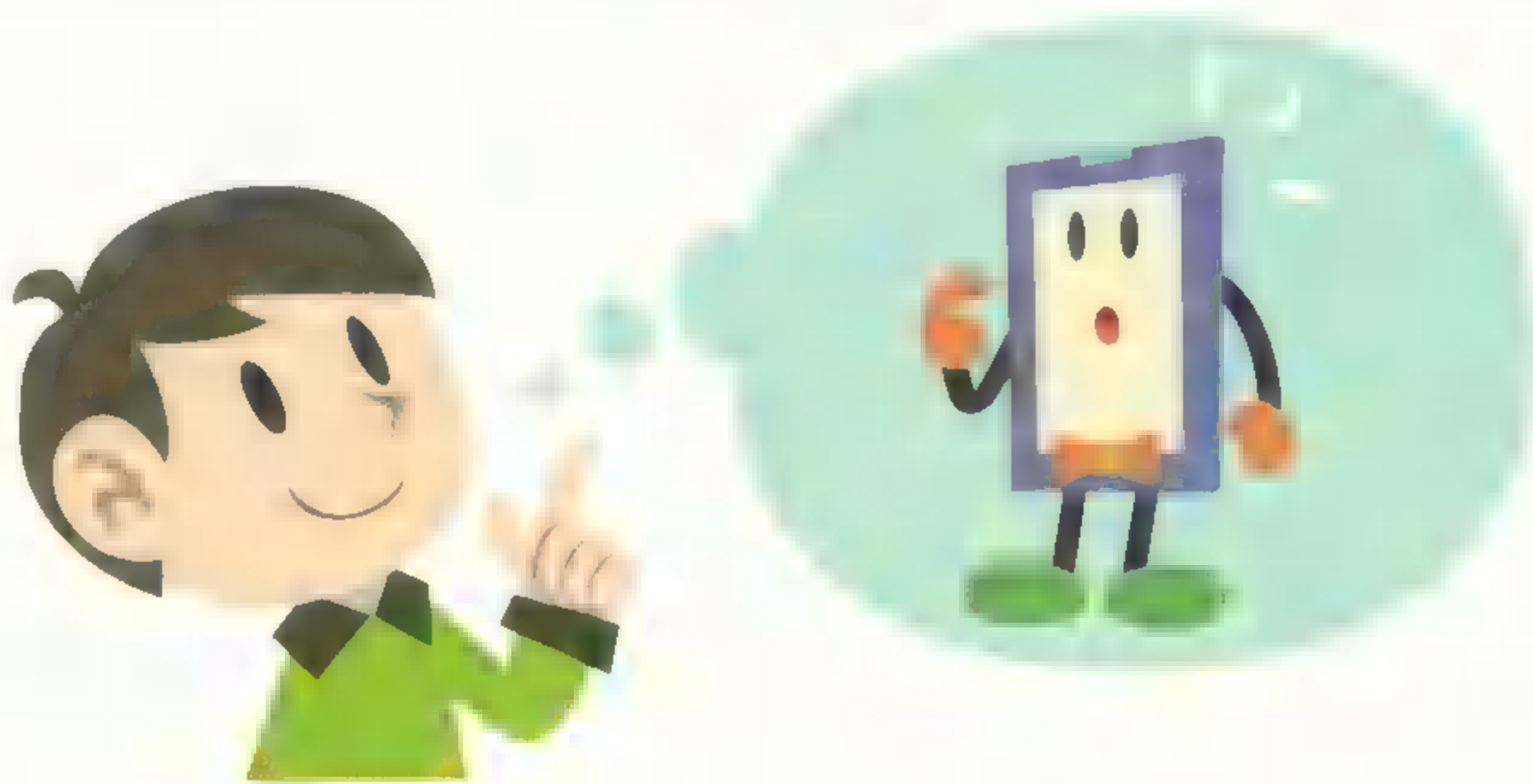
プログラミング言語

というものであるからです。プログラミング言語は、コンピュータのプログラムを書くための言葉です。プログラミング言語は、プログラムを作成するのに欠かせない技術となっています。このプログラムとは、いったいどのようなものなのでしょうか。

プログラム

プログラムは、コンピュータが「どんなふうに動くのか」という動作をあらかじめ設計したものです。このコンピュータとは、携帯端末などに含まれているコンピュータでもかまいません。

たとえば、旅行会社のWebサイトを開発することになったと考えてみてください。このとき、「大人3人分のバス代金額を計算する」などというコンピュータならではの処理をしたいとしましょう。

■コンピュータに仕事をしてもらいたいな・・・

このとき、コンピュータのために、次のような指示・手順を設計しておくのです。

■日本語で書いてみたプログラム

1人分のバス代を設定しておく
バス代を3倍する
合計金額を表示する

私たちはこうした指示をプログラムとして作成・準備します。プログラムを用意し、そのとおりにコンピュータを動かそう、というわけです。

プログラミング言語

さて、相手が人間であれば、「バス代を計算して！」という指示を伝えれば、すぐに意味をわかってもらえるでしょう。けれどもコンピュータにはそのままの指示では伝わりません。私たちはコンピュータにも伝わるような言葉で指示をしなければなりません。このとき使われる言葉が、プログラミング言語なのです。

人間の言語に英語や中国語、日本語…などの種類があるように、プログラミング言語にもたくさんの種類があります。JavaScriptはもっともシンプルなプログラミング言語のひとつとなっています。

■コンピュータに指示するためにいろいろな言葉がある

プログラミング言語はたくさんの種類があります。本書で学んでいくJavaScriptのほか、PHPなどと呼ばれる言語が有名です。いろいろなプログラミング言語を習得していくことで世界が広がります。

JavaScriptの場合は、バス代の計算について次のように指示をすることになります。

■ JavaScriptで書いてみたプログラム

```
var bus = 200;
var sum = bus * 3;
document.write(sum);
```

バス代を設定します
3人分の料金を計算します
合計額を表示します

いかがでしょうか？　くわしい意味はわからなくても、少しだけでもイメージがつかめたでしょうか？　少し難しく感じられるかもしれませんが、こうしたプログラミング言語を習得すれば、多彩なWebサイトを開発していくことができるようになります。Webサイトの開発に携わる人間の一人として力をつけていくことができるでしょう。

それでは、JavaScriptでWebサイトを開発する手法を学んでいきましょう！

JavaScriptとWebの歴史

もともとWebの世界では、Webページを記述する言語としてHTMLが広く使われてきました。HTMLは、1枚の動かない単純な文書としてのWebページを作成するために使われてきた言語です。JavaScriptはこうしたWebページに動きをつけるプログラミング言語として発展してきました。

ただし、現在の最新バージョンであるHTML5では、Webページに動きをつける強力な機能も加えられました。JavaScriptとあわせて使うと便利です。

本書では、JavaScriptを中心としたさまざまな技術を紹介していきます。幅広く技術を学んで、Webサイトを開発していけるようになりましょう。

1-2 ツールを使ってみよう

ツールを用意しよう

最初に開発をするために必要なツールを準備しましょう。JavaScriptで開発をするためには次のツールを用意する必要があります。

PC（パソコン）

まず、最初に必要となるものが、PC（パソコン）です。一般的なPCは、ノート型やデスクトップ型と呼ばれる形をしています。また、よく使われるPCの種類としてWindowsとMacがあります。

■ノート型PCとデスクトップ型PC

エディタ

次に必要となるものがエディタ（テキストエディタ）です。エディタは、文字の並び（テキスト）を保存するためのツールです。エディタで作成したファイルは、テキストファイルという文字の並びだけを保存したシンプルなファイルとなります。

Webページの基本は、HTML言語で記述されたテキストファイルです。このWebページを入力するためのツールがエディタなのです。

Macでは「テキストエディット」、Windowsでは「メモ帳」というエディタが標準的にインストールされています。

■ テキストエディット（Mac）

Sample.html — 編集済み

```
<!DOCTYPE html>
<html lang="ja">
<head>
<meta charset="utf-8">
<link rel="stylesheet" href="Sample.css">
<title>サンプル</title>
<script type="text/javascript">
function welcome()
{
   window.alert("ようこそ！ ログインしてください。");
}
</script>
</head>
<body>
<img src="Sample.jpg" onclick="welcome(1"/>
</body>
</html>
```

■ メモ帳（Windows）

Sample.html - メモ帳

ファイル(F) 編集(E) 書式(O) 表示(V) ヘルプ(H)

```
<!DOCTYPE html>
<html lang="ja">
<head>
<link rel="stylesheet" href="Sample.css">
<title>サンプル</title>
<script type="text/javascript">
function welcome()
{
   window.alert("ようこそ！ ログインしてください。");
}
</script>
</head>
<body>
<img src="Sample.jpg" onclick="welcome()"/>
</body>
</html>
```

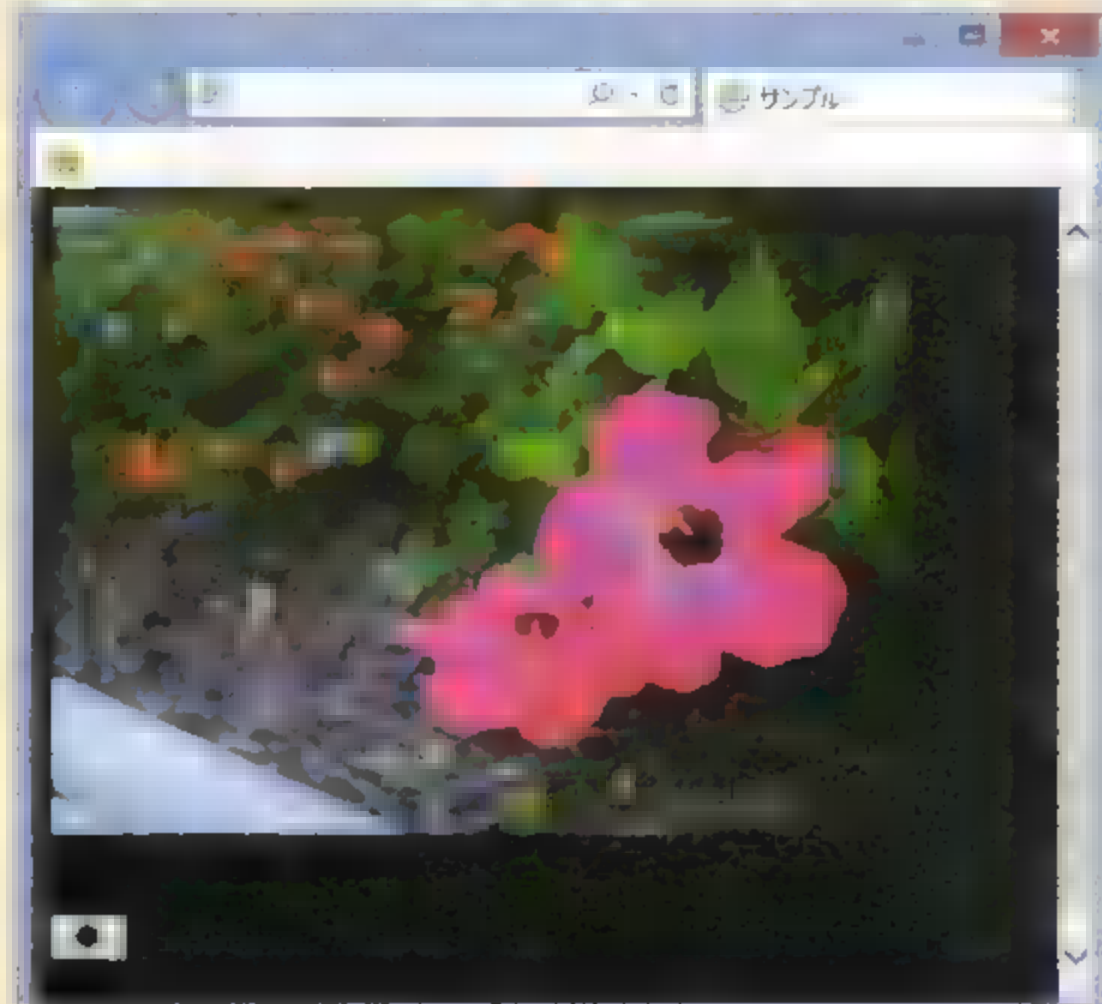
文章を作成する際によく使われている文書作成ソフト（ワープロ）は、文字の並び以外の文書レイアウト情報なども保存するため、プログラミングをしていくときには使いません。文字の並びだけを保存するエディタを使用するようにしてください。なお、Macのテキストエディットは、標準の設定ではレイアウト情報が保存されるため、あらかじめテキストファイルを扱うための環境設定をしておく必要があります。テキストエディットの環境設定の画面を開き、「フォーマット」オプションで「標準テキスト」を選択してください。また、エディタを組み込んだプログラミング専用の開発ツールを使うこともあります。さまざまな[illegible]の開発ツールが公開されています。

■ ブラウザ

最後に必要となるものがブラウザ（Webブラウザ）です。ブラウザは、エディタで作成したHTMLファイルをWebページとして表示してくれるツールです。「Safari」「Internet Explorer」「Google Chrome」「Firefox」などがあります。

■Safari (Mac)

■Internet Explorer (Windows)

Tips

ブラウザが動作するのはPC端末ばかりではありません。
スマートフォンやゲーム機などにもブラウザが用意されています。開発になれたら、さまざまな端末上のブラウザも使いこなしていくことができるでしょう。

JavaScriptを入力してみよう

JavaScriptは、大変シンプルで扱いやすい言語です。お使いのPCにインストールされているエディタとブラウザだけで開発することができます。

そこで試しに、JavaScriptで簡単なプログラムを作成してみましょう。お使いのPC上でエディタを起動し、次の文字を入力してみてください。英字は半角を使います。

Sample.html ■ はじめてのJavaScriptプログラム

```
<!DOCTYPE html>
<html lang="ja">          ← 半角英数字を使ってください
<head>
<title>サンプル</title>
<script type="text/javascript">          ← スラッシュです
function welcome()
{
  window.alert("ようこそ！ ログインしてください。");          ← セミコロンをつけます
}
</script>
</head>
```

```
<body>
<img src="Sample.jpg" onclick= "welcome()"/>
</body>
</html>
```

Tips

このプログラムの「window.alert・・・」の前では、行頭にスペースを入れて字下げを行っています。これは、{ }で囲まれた部分を読みやすくするためのプログラム上の工夫です。スペースキーまたはタブキーを押すと字下げすることができます。

入力が終わったらファイルを保存してください。「Sample.html」という名前で保存してみましょう。同じフォルダに、デジタルカメラなどで撮影した写真ファイルを、「Sample.jpg」のファイル名で保存してください。

Tips

なお、HTMLファイルを保存する際に文字コードには気を付ける必要があります。Webでは、一般的には文字コードとして「utf-8」が使われます。ただし。Windowsのメモ帳の標準設定では「Shift_JIS」で保存されます。また、Webページが文字化けする場合には、3行目の<head>の下などに、<meta charset="お使いの文字コード">と記述して、文字コードを指定する必要があります。utf-8なら、<meta charset="utf-8">になります。

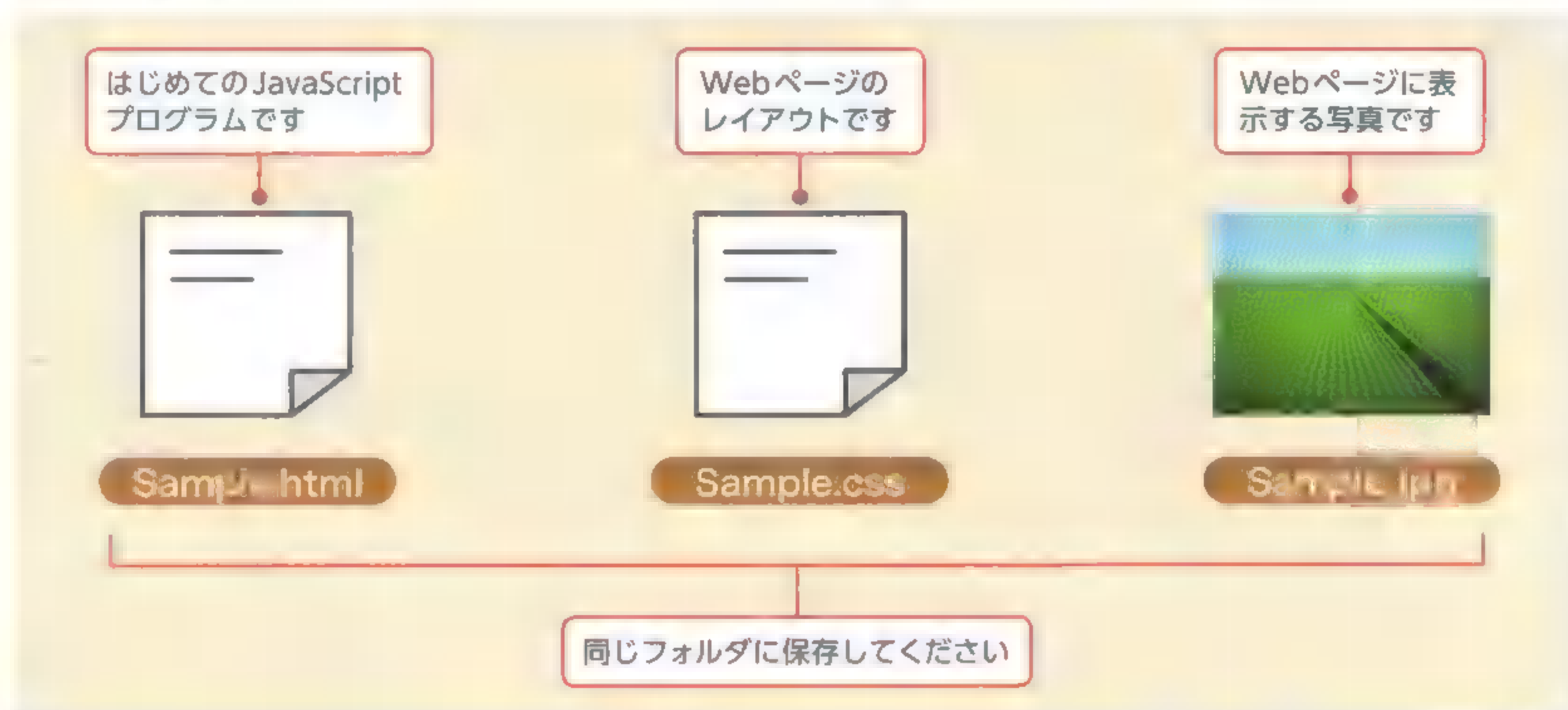
また、次のようなテキストファイルも作成します。このファイルの名前は「Sample.css」としてください。Webページのレイアウトを担当するファイルです。このファイルも同じフォルダに保存してください。

Sample.css ■ Webページのレイアウト

```
body{
  background-color: #000000;
  color: #FFFFFF;
}
```

Webページをレイアウトしています

したがって、次の３つのファイルを作成することになります。

■作成するファイル

「.html」や「.jpg」、「.css」のように、ピリオド（.）のあとにつける名前を「拡張子」と呼びます。一般的にコンピュータの世界では、ファイルの種類を区別するために使われます。「.html」は、HTMLによって書かれたファイルであることをあらわしているのです。PCの設定によっては、拡張子を表示しないようになっている場合もあります。

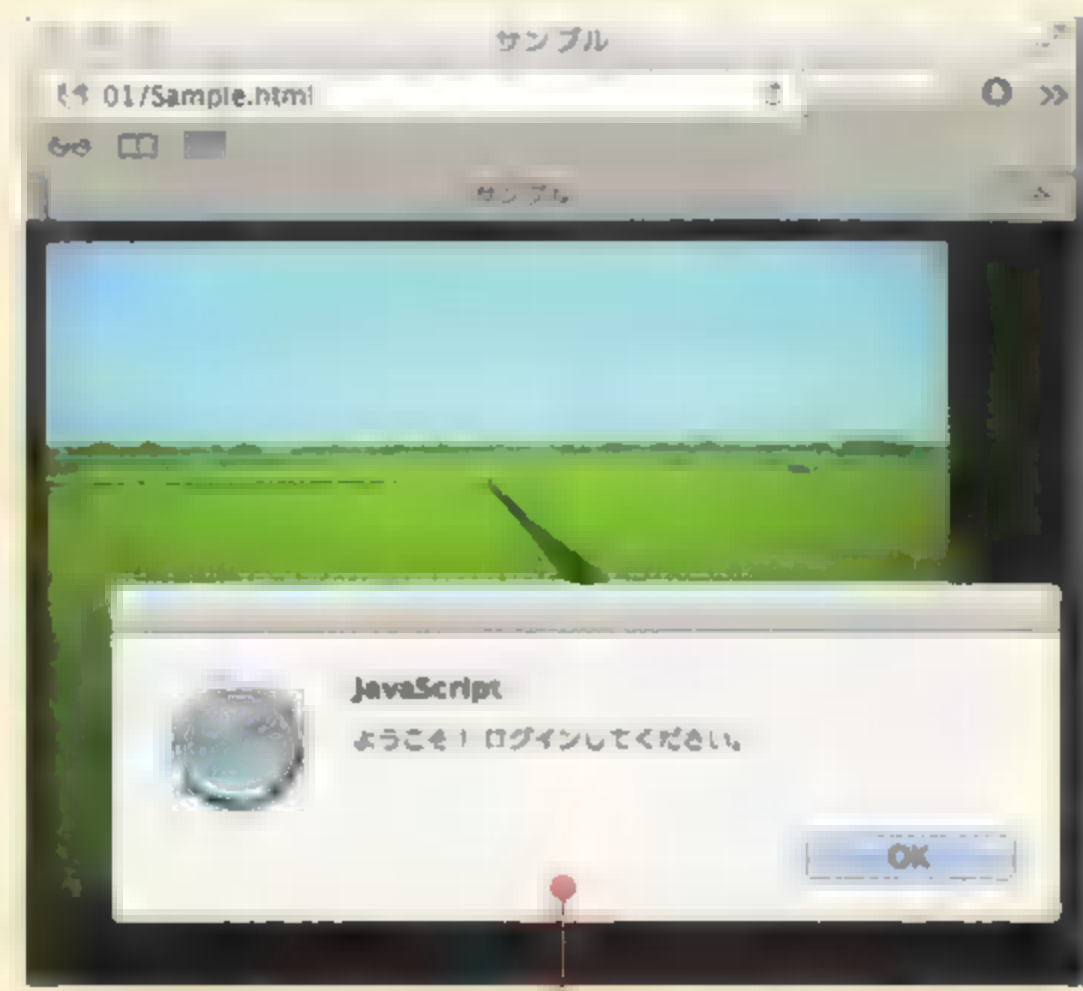
JavaScriptを動かしてみよう

すべて作成したら、Sample.htmlをあらわすファイルのアイコンをダブルクリックしてみてください。ブラウザが起動し、Webページが表示されます。

ここで画像をクリックします。すると、次図のようにメッセージが表示されるでしょう。

いかがでしょうか？　思いどおりに動いたでしょうか。

こんなふうにJavaScriptでWebページを動かすことができるのです。まずはJavaScriptのプログラムを体験してみてください。

■ JavaScriptで動かす

JavaScriptのプログラムが実行されるとき、ブラウザによってはユーザーの許可を行うように設定されている場合があります。許可を行ってプログラムを実行してみてください。

1-3 しくみはどうなってるの？

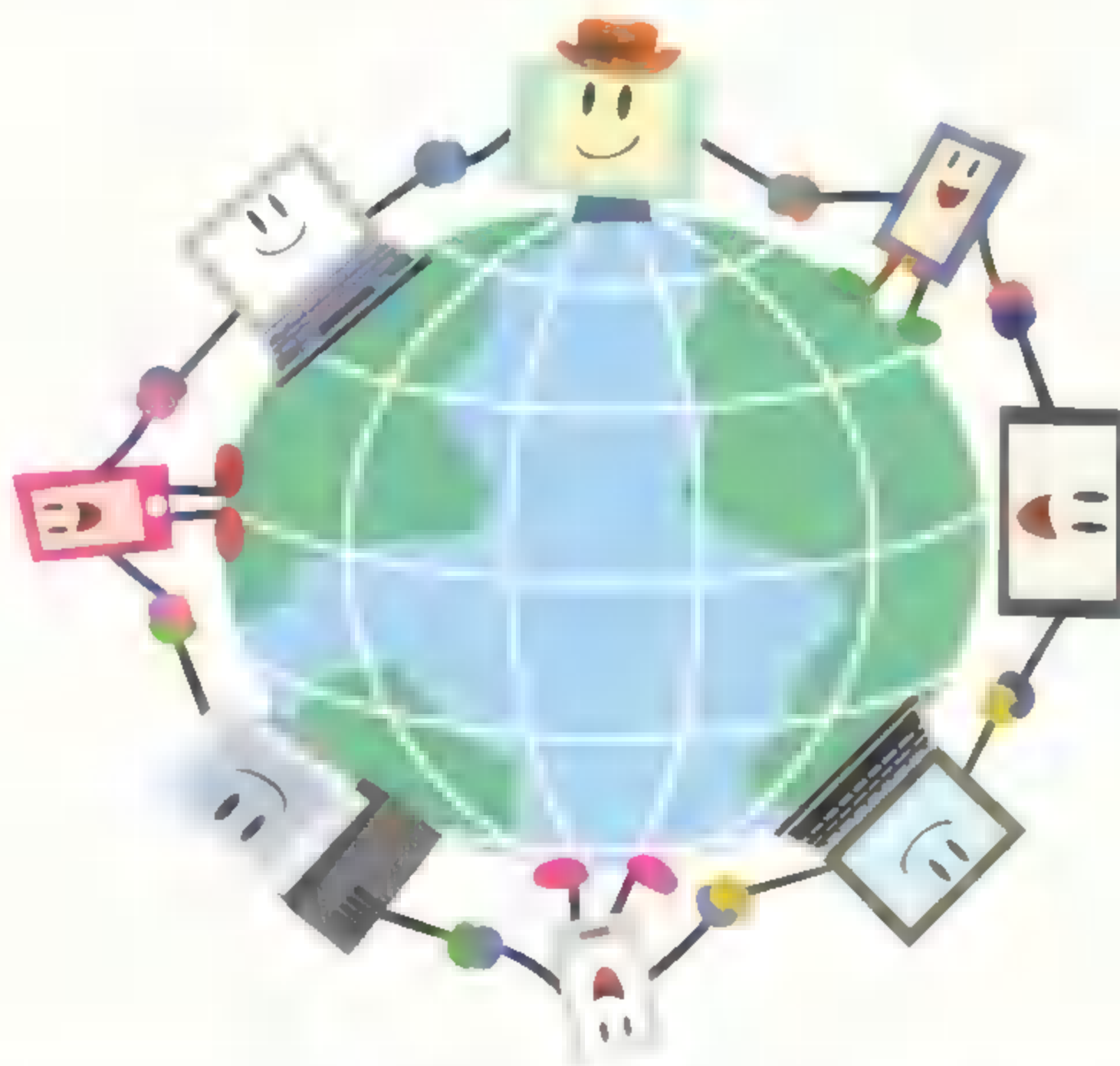
Webのしくみを知ろう

JavaScriptは、Webサイトを構築する際によく利用されるプログラミング言語です。そこでこの節では、「Web」についてよく知っておくことにしましょう。

Webは最新のニュースを見たり、キーワード検索をして情報を調べたり、買い物をして商品をカートに入れたりするために使われています。Webは私たちにとってとても身近で利用しやすい情報ツールとなっています。

インターネット

Webはインターネットを経由して配信されます。インターネットは世界中のコンピュータをつなぐ巨大なネットワークです。私たちはWebページを公開したり、インターネットに接続してWebを見たりすることができます。

■世界に広がるインターネット

サーバー

Webページをまとめ、Webサイトとして公開するコンピュータをサーバーといいます。サーバーは、ユーザーからのリクエストに応じてWebページの情報を配信します。Webサイトには、数多くのユーザーがアクセスすることになります。このため、サーバーにはPCをはじめ、処理能力の高いコンピュータが使われています。

■ Webサイトを公開するサーバー

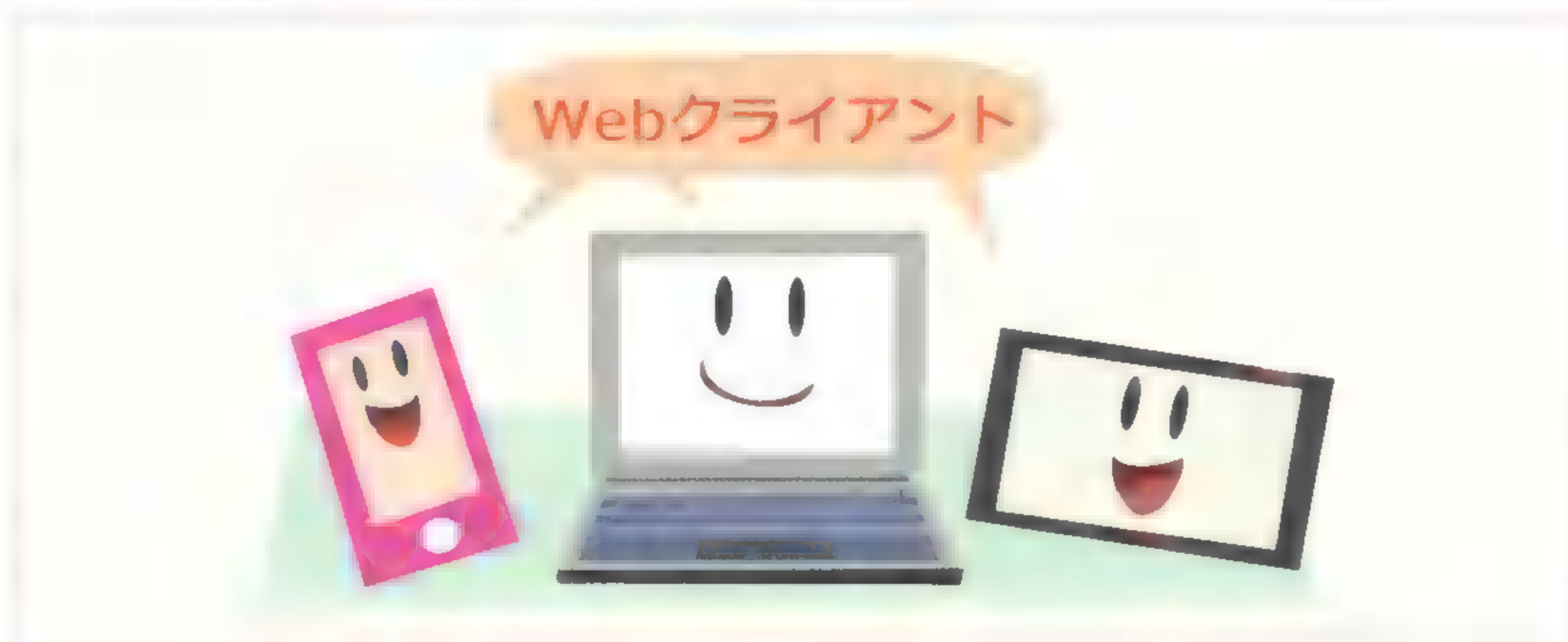
Tips

サーバーでWebサイトを公開する場合には、さまざまなセキュリティ管理が必要です。サーバーが別の場所でほかの管理者によって管理されている場合は、FTPなどと呼ばれるツールを使って、必要なファイルをサーバーに送信してからWebサイトを公開する場合もあります。
Webサーバーの管理方法やFTPによるファイルの送信方法については、各種書籍・ツールのヘルプなどの情報を参照してください。

クライアント

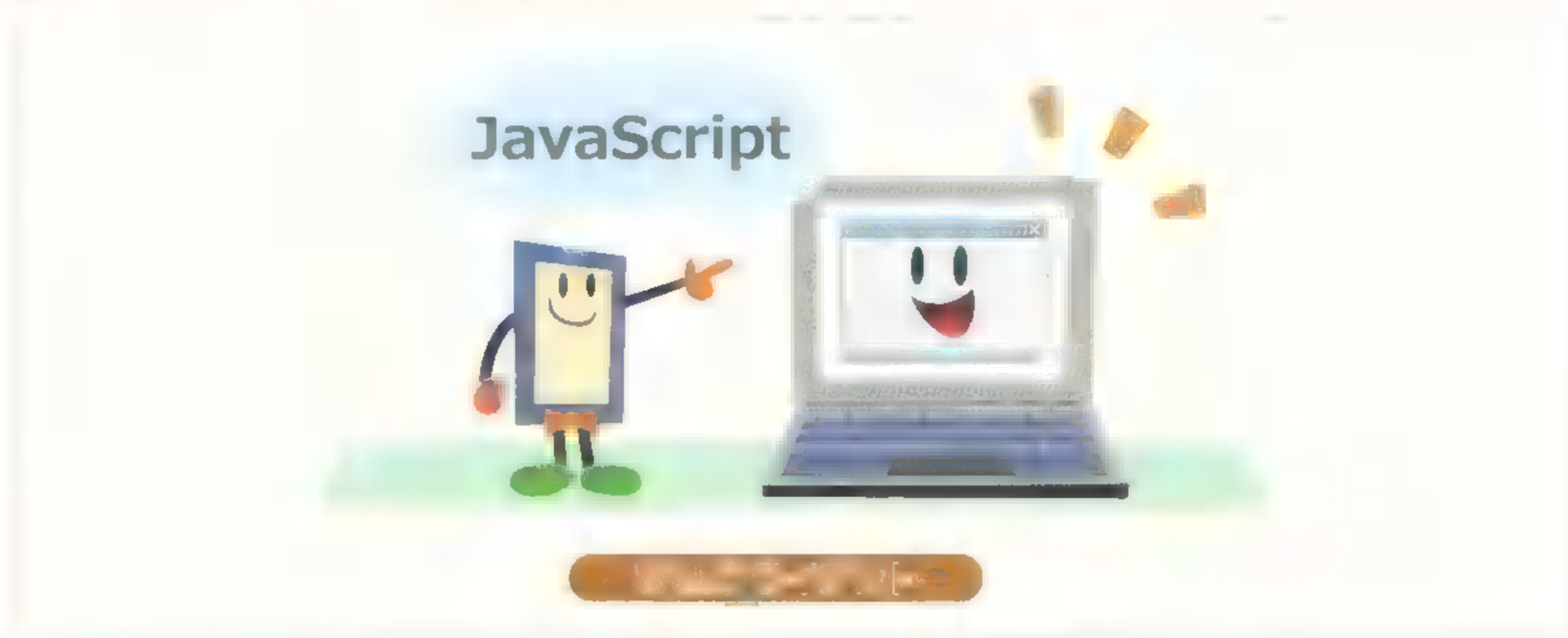
Webサイトを閲覧するための端末をクライアントといいます。クライアントは、ブラウザが動作する端末であればよいことになります。

PCやスマートフォン、ゲーム機など、ブラウザが動作する端末として。さまざまなクライアントが使われています。また、Webサイトを見るブラウザ自体を「クライアント」と呼ぶ場合もあります。

■Webページを閲覧するためのクライアント

JavaScriptはクライアントで動く

ところで、JavaScriptで書かれたプログラムを実行するのはブラウザです。つまり、JavaScriptのプログラムは、クライアント上で動作しているプログラムであるということになります。

これは、JavaScriptによるプログラムの動作をイメージするうえで、とても大切なことです。おぼえておくと役に立つでしょう。

■JavaScriptはクライアント上で動作する

JavaScriptは、クライアント上で動作するプログラミング言語です。これに対して、サーバー上で動作する代表的な言語としてPHPと呼ばれる言語などがあります。どのコンピュータでプログラムが動作しているのか、プログラムの動作環境をイメージすることは、プログラミングにとってとても大切です。

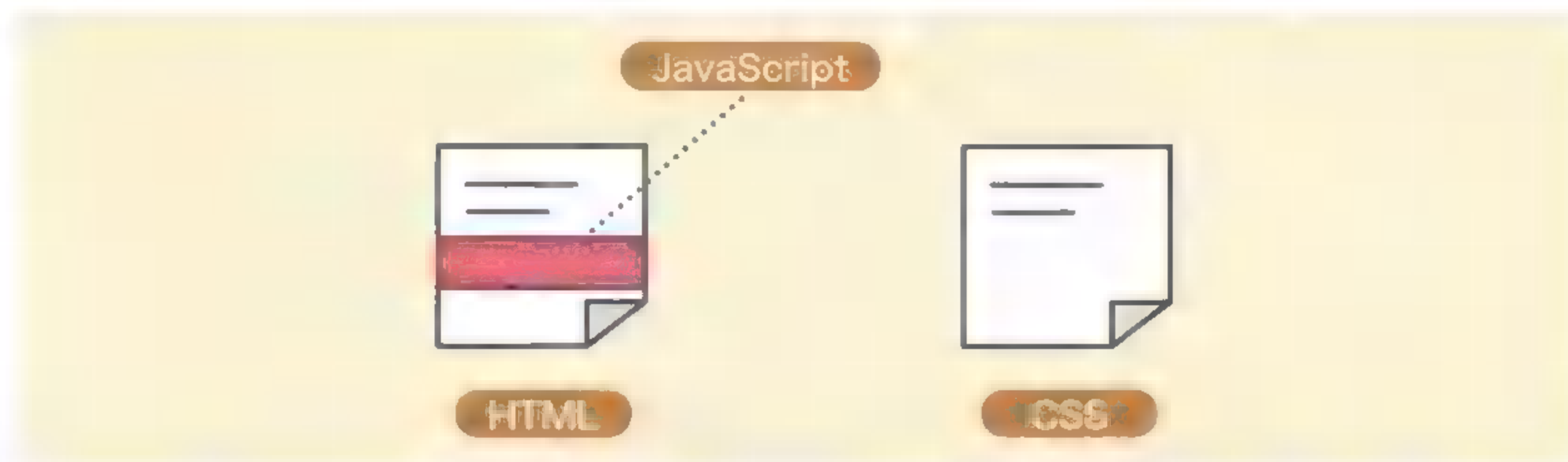
1-4 ファイルを用意しよう

必要なファイルをおぼえておこう

今度は、Web上のプログラムを構成するために必要なファイルや要素をおさえておきましょう。もっとも基本的なWebプログラミングのための構成要素は、次のようになっています。

- HTML
- JavaScript
- CSS

これらはどれもエディタを起動して文字の並びを入力し、ファイルとして作成していくことになります。どのようなものか、1つずつみていきましょう。

■Webプログラミングに必要な要素

HTML

HTMLはWebページを書き表すための言語です。< >や</ >という形をしたタグと呼ばれる記号ではさんで、ページの項目や内容を入力・作成することになっています。

たとえば、タイトルは次の図のように<title>〜</title>タグではさんで書きます。はさまれた部分がWebページのタイトルとなるわけです。

タグにはさまれた部分を要素と呼びます。「title」要素は、Webページのタイトルをあらわす要素です。

■HTMLのタグ

要素の中にも要素を書くことができます。たとえば、一番外側の要素は、HTMLであることをあらわす「html」要素とします。この「html」要素の中に、タイトルとなる「title」要素や内容となる「body」要素を書くことになります。

HTMLのタグは、大文字・小文字どちらでもかまいません。ただし、現在ではタグを小文字で書くことが一般的になっています。

HTMLの要素

Column

HTMLの主な要素には、次の表にあげているものがあります。このほかにもWebページを記述するために、数多くの要素が用意されています。

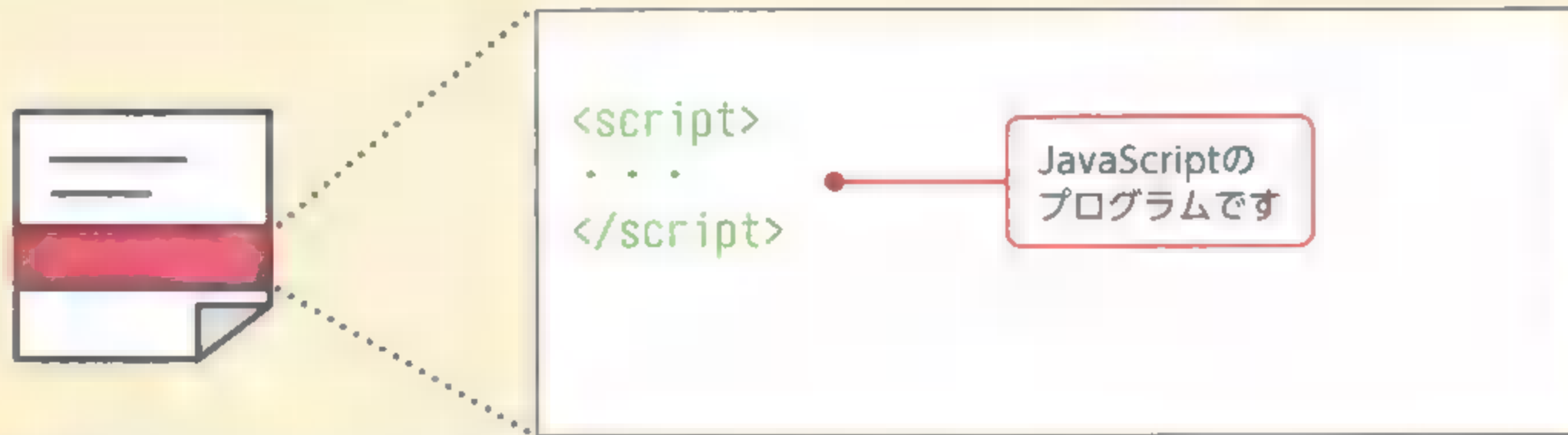
■HTMLの主な要素

要素	説明
h1～h6	見出し
header	ヘッダ
footer	フッタ
article	記事
section	セクション
aside	副次的な内容
nav	ナビゲーション
address	連絡先
p	段落
div	範囲
img	画像
table	表
form	フォーム

JavaScript

JavaScriptはさまざまな書き表し方があります。そのうち、もっとも簡単な方法として、HTMLのファイルの中に、HTMLと一緒に書いておく方法が利用されています。

HTMLファイルの中でJavaScriptをあらわすには、HTMLの「script」要素を使います。このタグにはさんで、JavaScriptのプログラムを入力するのです。

■ JavaScriptはHTMLファイルの中で「script」要素に囲んで記述する

JavaScriptの入力場所

Column

JavaScriptをあらわす「script」要素は、HTML中のさまざまな場所に書くことができます（❶）。

HTML文書中の「head」要素内（❷）のほか、「body」要素内（❸）や、HTMLファイルとは別のファイル（拡張子は「.js」）として作成しておくこともできます。なお、先ほどの1-2節の例では、「head」要素の内部に書きました（❷の例）。

❶

```
<html>

<script>
</script>

<script>
</script>

</html>
```

❷

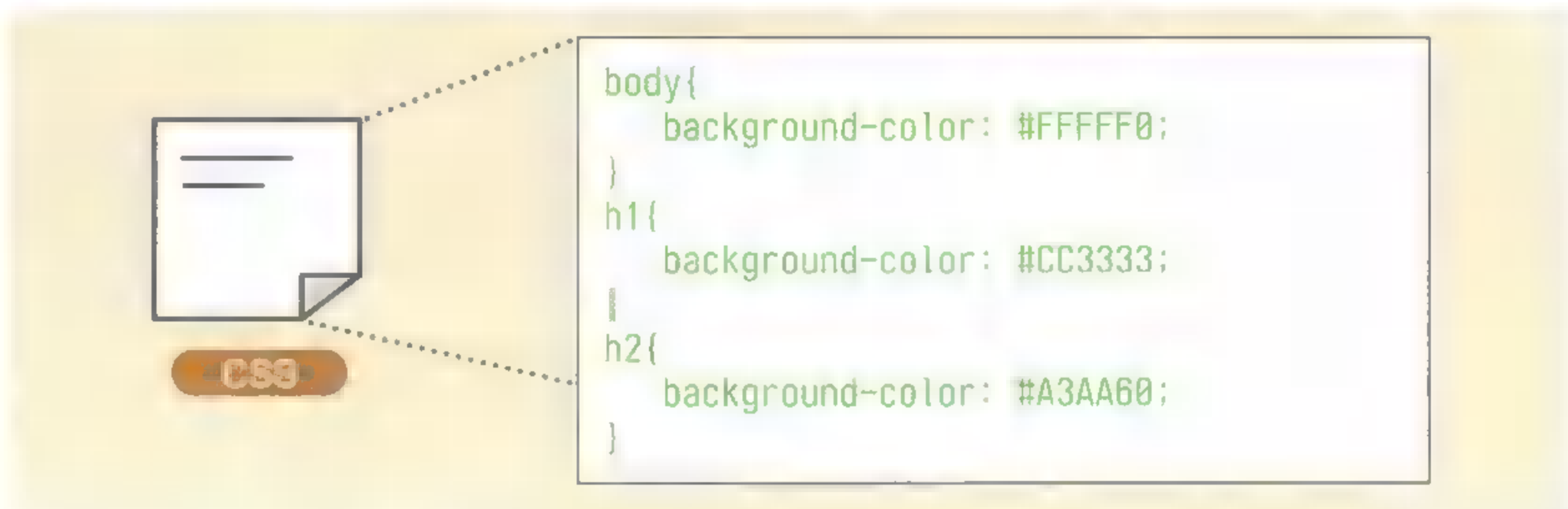
```
<html>
<head>
<script>
</script>
</head>

</html>
```

❸

```
<html>

<body>

<script>
</script>
</body>
</html>
```

CSS

もう1つ、Webページを作成する際に重要な要素がCSS（カスケーディングスタイルシート）です。CSSはWebページのデザインやレイアウトを決めるために使われるファイルで、単純にスタイルシートとも呼ばれています。通常、HTMLとは別のファイルとして作成し、HTMLファイルの中にCSSファイルの名前と場所を指定します。

■デザイン・レイアウトを受けもつCSSファイルはHTMLとは別ファイルにする

CSSでは次のような形でレイアウトを指定します。

構文

セレクタ {プロパティ：値}

セレクタは、HTML文書中の要素などをあらわします。プロパティは、その要素の色やフォントなどの属性です。HTML文書中の要素の色やフォントなどの値を指定して、レイアウトを作成するのです。たとえば、次の指定は「body」要素の背景色を指定するものです。

■スタイルシートの指定方法

ファイルの場所を指定するには

Column

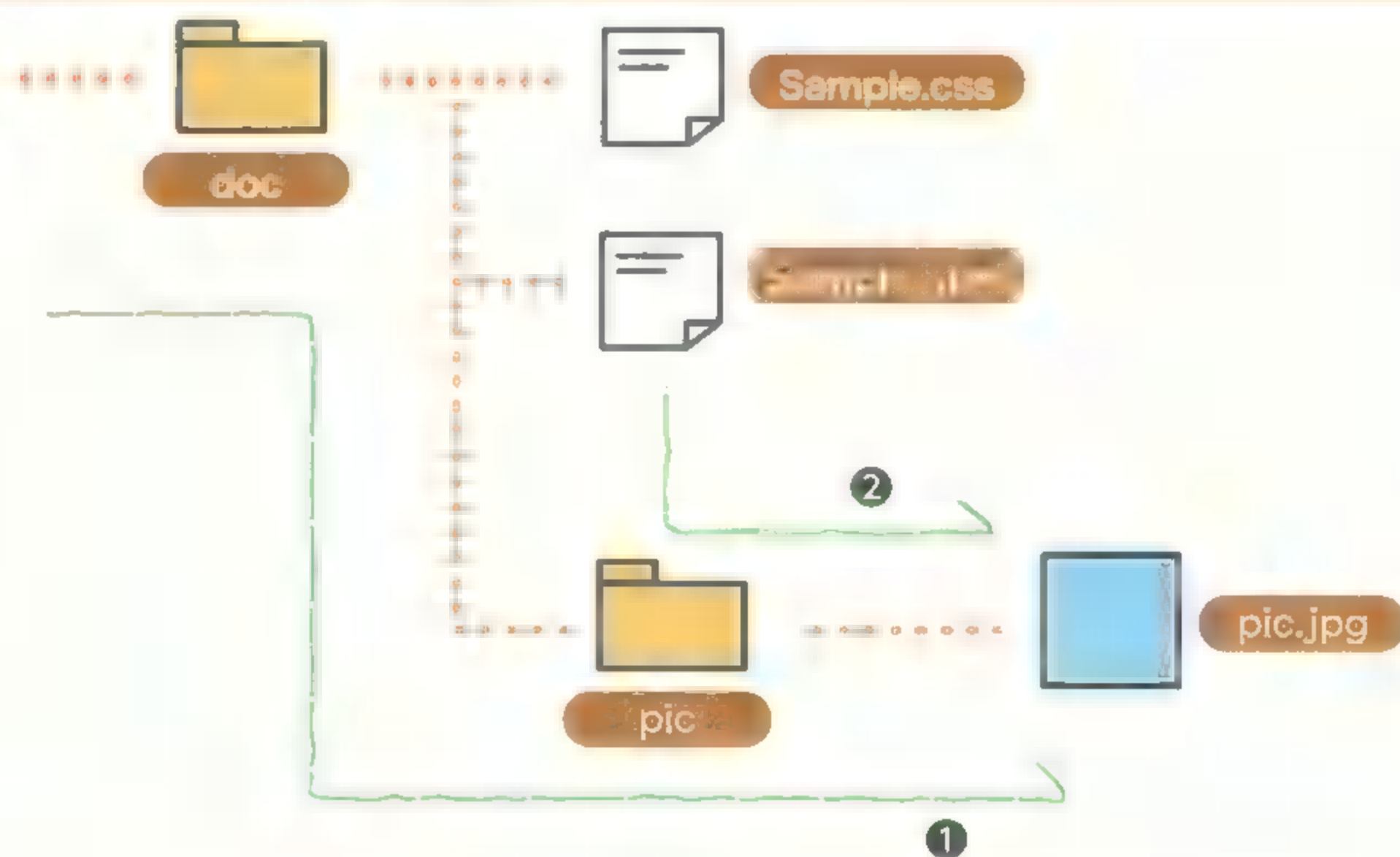
このほかにもWebサイトの構築にあたっては、画像ファイルや音声・動画ファイルなどを準備する必要があります。HTMLやJavaScriptの中では、これらのファイルの場所を指定することがあります。

ファイルの場所を指定するには、URLという書き方を使います。そこで、URLの書き方について少し紹介しておきましょう。

URLには、どこからでも同じようにあらわす方法（❶）と、指定するHTMLファイルの場所から指定する方法（❷）があります。

たとえば、図中のようにpicフォルダに保存した画像ファイルについて考えてみましょう。この画像をSample.html中で指定するには、❶の方法では「http://sbcr.jp/doc/pic/pic.jpg」と指定します。また、❷の方法では、HTMLの場所からたどって「pic/pic.jpg」と指定します。

なお、先ほどの1-2節のように同じフォルダ内に保存したファイルについては、ファイル名だけで指定できます。

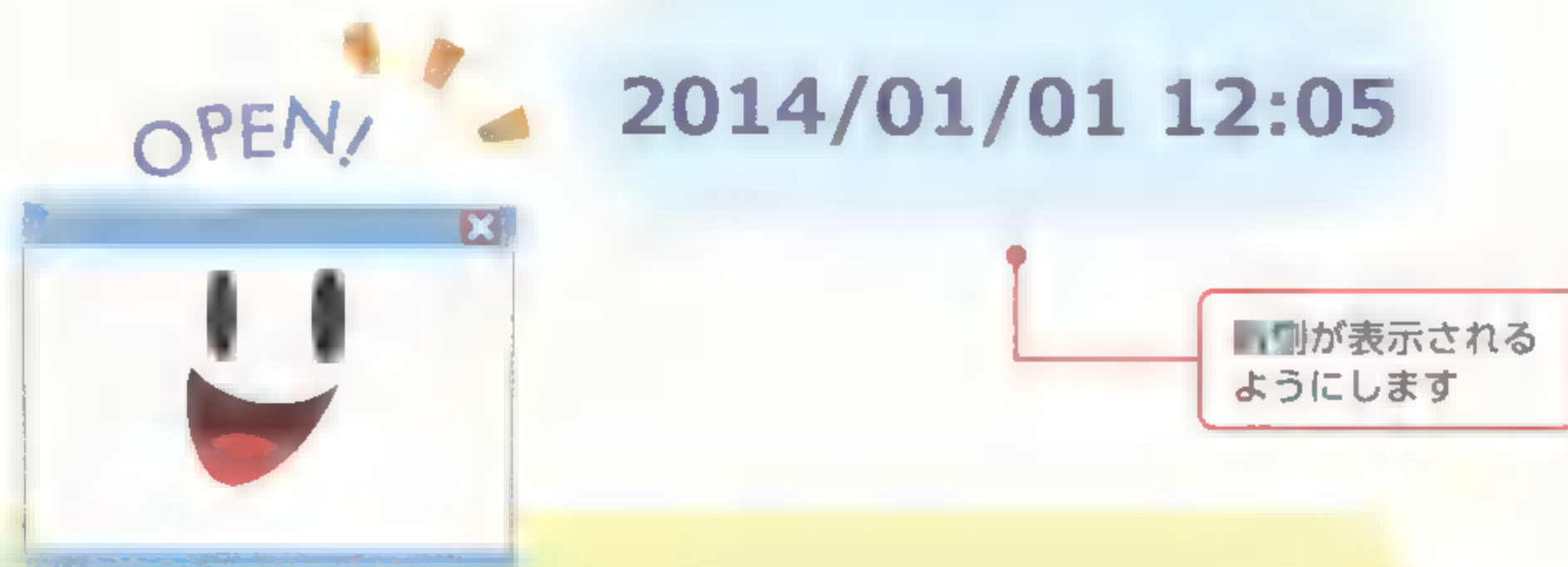
こうしたファイルの指定方法についておぼえておくと便利でしょう。

2-1 時刻を表示しよう

チャレンジ! 時刻を調べよう

この章では、JavaScriptの中で、もっとも基本となる事柄について学んでいきましょう。この節では、JavaScriptを使い、ユーザーがWebページを開いた「時刻」を表示してみることにします。

通常、HTMLの機能を使うだけでは、ユーザーがWebページを開いた時刻をWebページに表示することはできません。もともとHTMLは、Webページに表示する内容をあらかじめ書きあらわすためだけに使われてきた技術だからです。

けれどもJavaScriptの機能を使えば、Webページに高度な機能を付け加えることができます。ユーザーがWebページを開いた時点の時刻を表示することができるのです。

■ Webページを開いた時刻を知る

時刻を表示するプログラム

それでは、時刻を表示するプログラムを作成してみましょう。プログラムを次のように入力していきます。エディタを起動し、HTMLファイルを作成します。<script>〜</script>タグの間にJavaScriptのプログラムを入力してみてください。プログラミングの世界では、こうした入力プログラムをコードと呼ぶ場合があります。

Sample2-1-1.html ■ 時刻を表示する

```
<!DOCTYPE html>
<html lang="ja">
<head>
<link rel="stylesheet" href="Sample.css">
<title>サンプル</title>
</head>
<body>
<script type="text/javascript">
var d = new Date();
document.writeln(d);
</script>
</body>
</html>
```

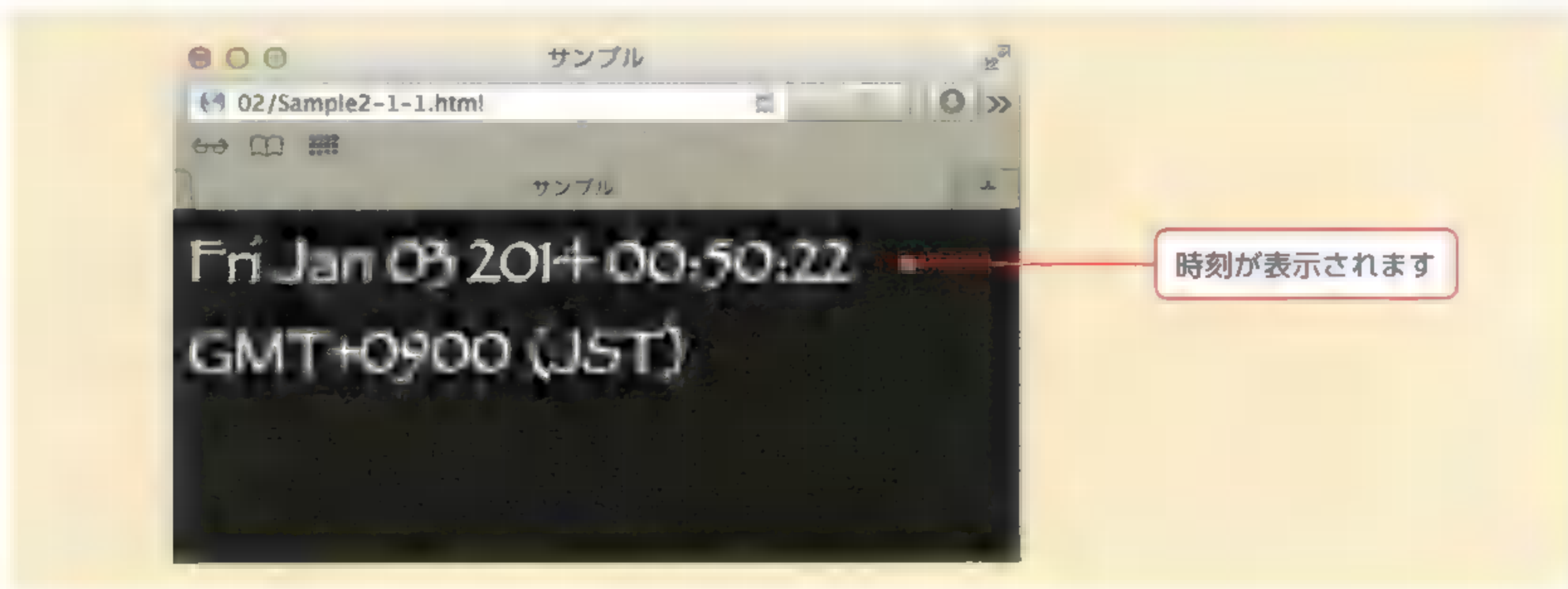
時刻を調べます
時刻を表示します

なお、この章では次のスタイルシートを使います。次のCSSファイルを作成し、HTMLファイルと同じフォルダ内に保存しておいてください。

Sample.css ■ スタイルシート

```
body{
   background-color: #000000;
   color: #FFFFFF;
   font-size: 2em;
   font-family: fantasy;
}
#on{
   background-color: #2e8b57;
   color: #FFFFF0;
   font-family: fantasy;
}
#off{
   background-color: #000000;
   color: #FFFFF0;
   font-family: fantasy;
}
```

さて、作成したHTML文書をダブルクリックして開くと、次のようなWebページが表示されます。ページを開いた時刻が、Webページに埋め込まれて表示されているのがわかることでしょう。

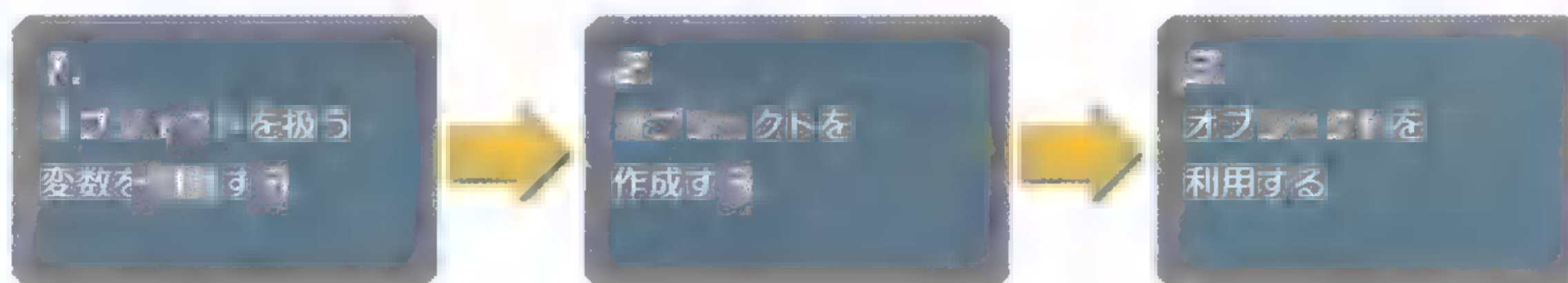
■Sample2-1-1の表示

順番に処理される

「script」要素を使ってJavaScriptのコードを記述すると、Webページを開いたときに、「script」要素内の処理が順番に処理されるようになっています。

JavaScriptでは、処理の単位を文と呼んでいます。文の末尾には;（セミコロン）をつけます。つまり、ここでは<script>〜</script>に囲まれた、次の3文の処理が順番に行われるわけです。

オブジェクトを利用する

それでは、このJavaScriptでは一体どんな処理をしているのでしょうか？　ここで少しみておくことにしましょう。

JavaScriptでは、オブジェクトというものを作成して利用します。オブジェクトは、次のような順で利用することになっています。次の手順を、順番を追ってみていきましょう。

オブジェクトを扱う変数を用意する

JavaScriptではオブジェクトを扱うために、まず変数とよばれるしくみを用意します。変数はプログラミング言語でよく使われるしくみです。次のように変数の名前を用意して使います。

構文

```
var 変数の名前;
```

ここでは、次のように変数の名前を用意しているのです。

```
var d;
```

dという名前の変数を用意しています

varで変数を準備することを示します。dは自分で用意する変数の名前なので、原則として適当な文字列を使ってかまいません。ここでは、日時であることをあらわすためにdという名前にしてみました。変数を準備するイメージをつかんでみてください。

■ 変数を準備する

Tips

変数の名前には、次のような文字を使ってください。

- 文字・数字・アンダースコアなどを用います。
- 数字ではじめることはできません。
- 大文字と小文字は異なるものとして区別されます。
- あらかじめJavaScriptが予約している「キーワード」を使用することはできません。主なキーワードとして、varやfunctionがあります。
- 長さには制限がありません。

たとえば、「a」や「abc」などを使うことができるでしょう。数字ではじまる「123a」などは使うことができません。

オブジェクトを作成する

次にオブジェクトを作成します。オブジェクトは、newというキーワードで作成します。newは「『新しく』作成する」という意味ですので、わかりやすいことでしょう。

オブジェクトは、プログラムに必要なデータや機能をまとめたものです。ここでは、日付に関する機能をまとめた、Dateオブジェクトという種類のオブジェクトを作成しています。

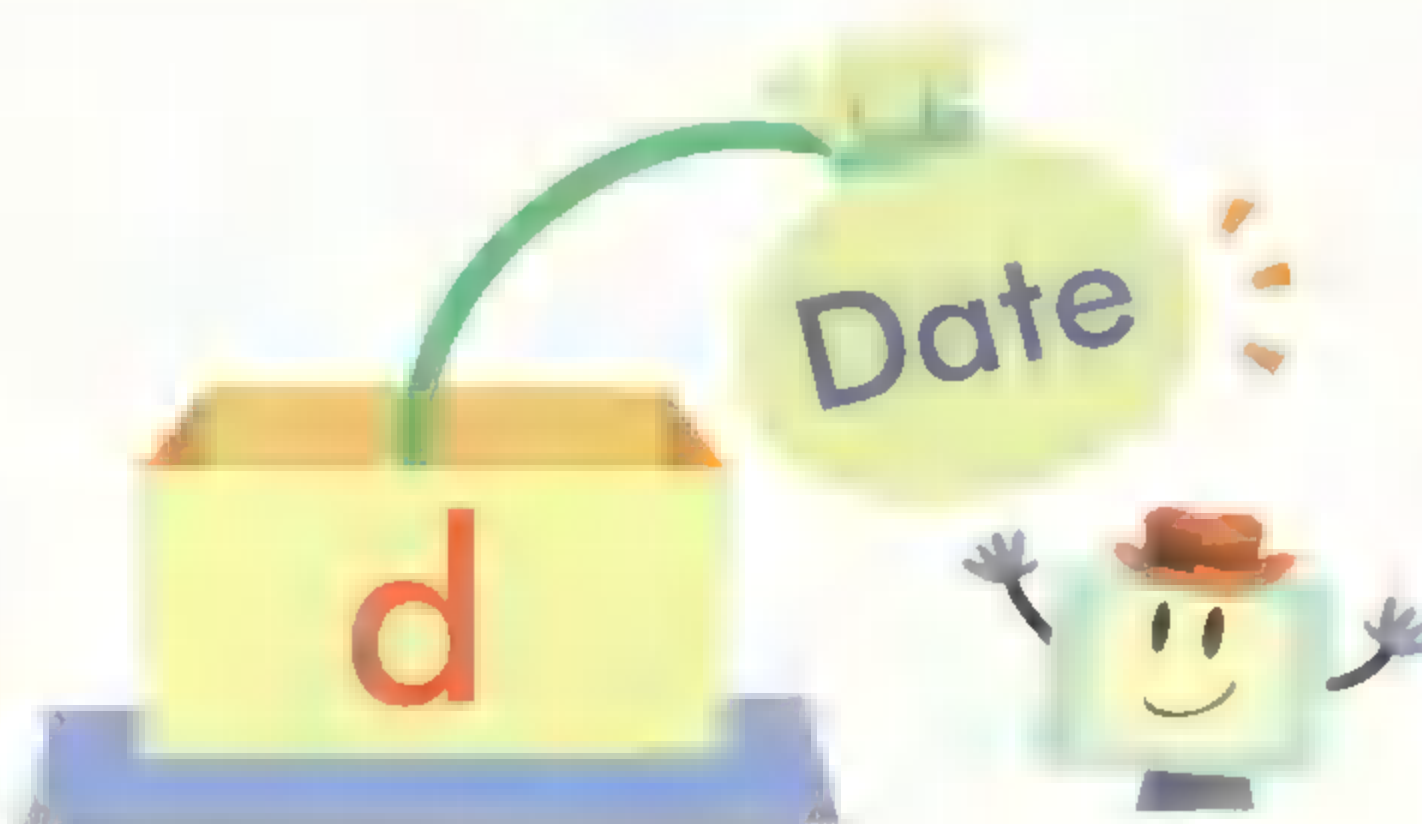
オブジェクトを作成したら、＝という記号を使って、変数の名前に設定します。

これで、dという変数の名前で作成したオブジェクトを扱うことができるようになります。

つまり、次のようにしてオブジェクトを作成したわけです。

構文

```
変数の名前 = new オブジェクト名();
```

■オブジェクトを作成する

これまで行った変数の準備とオブジェクトの作成は、次のようにまとめて簡単に書くこともできます。

```
var d = new Date();
```

これで作成したDateオブジェクトを、変数dですぐに扱うことができるようになります。本書ではこれから、この記述方法を使っていくことにしましょう。

オブジェクトを利用する

オブジェクトを作成したら、最後に変数を指定してオブジェクトを利用します。ここでは、( )の中にオブジェクトを扱っている変数を指定することで、日時を表示しています。これでオブジェクトを利用することができました。

```
document.writeln(d);
```
時刻を表示します

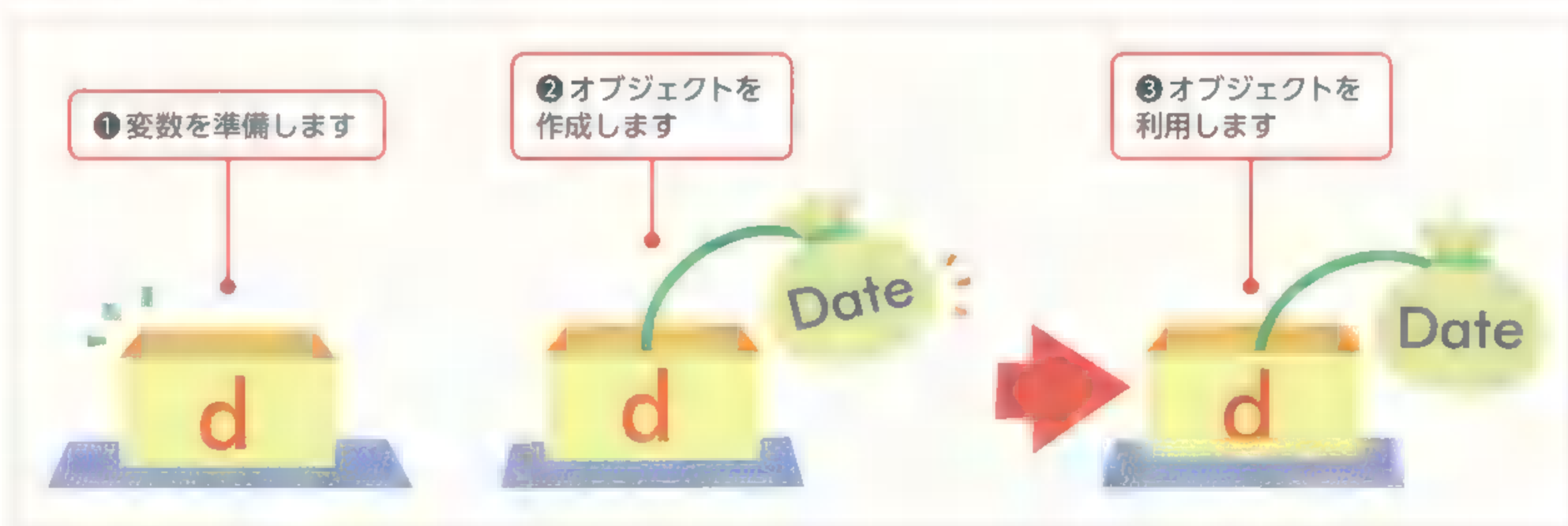
■オブジェクトを利用する

Tips

オブジェクトは、データと機能をまとめたものとなっています。つまりDateオブジェクトは、日時データとそのデータを調べたり設定したりする機能をまとめたものです。日時オブジェクトを利用することで、日時を調べることができるわけです。オブジェクトの利用方法については、これから少しずつみていきましょう。

さて、オブジェクトを利用するための3つの手順を理解することができたでしょうか？ もう一度3つの手順をふりかえってみておいてください。

■オブジェクトを利用する手順

オブジェクトの[illegible]

JavaScriptには、Dateオブジェクトのほかにも、次のような基本のオブジェクトが用意されています。さまざまなオブジェクトがあり、利用することができます。

また、プログラム上よく使われる機能を、新しいオブジェクトとしてまとめて設計することもできます。高度な利用方法も可能となっているのです。

■オブジェクト

オブジェクトの[illegible]	説明
Objectオブジェクト	オブジェクトに関する機能
Arrayオブジェクト	配列に関する機能
Booleanオブジェクト	[illegible]に関する[illegible]
Errorオブジェクト	エラーに関する機能
Eventオブジェクト	イベントに関する[illegible]
Dateオブジェクト	日付に関する機能
Functionオブジェクト	関数に関する機能
Mathオブジェクト	数学に関する機能
Numberオブジェクト	数値に関する機能
RegExpオブジェクト	正規表現に関する[illegible]
Stringオブジェクト	文字列に関する機能

2-2 オブジェクトを使いこなそう

チャレンジ！ オブジェクトを活用しよう

オブジェクトは便利なものです。オブジェクトには、データや機能がまとめられています。前の節で使ったDateオブジェクトの場合は、日時データとそのデータを調べたり設定したりする便利な機能がまとめられています。そこでこの節では、オブジェクトをもっと活用してみることにしましょう。

■オブジェクトの機能

オブジェクトの機能を利用するには？

オブジェクトにまとめられているデータは、プロパティと呼ばれています。また、オブジェクトにまとめられた機能は、メソッドとよばれています。プロパティとメソッドは、まとめてメンバと呼ばれることもあります。

たとえば、Dateオブジェクトには、主なものとして次のメンバがあります。日時に関するさまざまな機能を、メンバとして利用することができるのです。

■Dateオブジェクトの主なメンバ

メンバ	説明
getFullYear()	年を取得する
getMonth()	月を取得する
getDate()	日を取得する
getDay()	曜日を取得する
getHours()	時を取得する
getMinutes()	分を取得する
getSeconds()	秒を取得する
getMilliseconds()	ミリ秒を取得する
setFullYear(year)	年を設定する
setMonth(month)	月を設定する
setDate(date)	日を設定する
setDay(day)	曜日を設定する
setHours(hour)	[illegible]を設定する
setMinutes(min)	分を設定する
setSeconds(sec)	秒を設定する
setMilliseconds(msec)	ミリ秒を設定する
toDateString()	日付文字列を取得する
toTimeString()	時刻文字列を取得する
toLocaleDateString()	ローカル日付文字列を取得する
toLocaleTimeString()	ローカル時刻文字列を取得する
toString()	文字列を取得する

日付に関するさまざまな情報を得ることができるでしょう。オブジェクトのメンバは、次のようなJavaScriptのコードを入力することで利用できるようになっています。

たとえば、オブジェクトをdという変数で取り扱っている際に、「年」を調べる機能（getFullYear()）を利用するには、次のようなコードを作成するわけです。

```
d.getFullYear();
```

年を調べる[illegible]を利用できます

メソッドは「メソッド名()」というように( )をつけて使います。ここでは( )の中には何も指定していませんが、( )の中には機能のために必要な情報を指定することがあります。たとえば、日付を表示する際に使ったdocument.writeln()は、Documentオブジェクトのwriteln()メソッドです。document.writeln("文字列")という形式で使っています。( )内に表示する文字列情報を指定しているわけです。

応用! さまざまな日時を表示してみよう

それでは、Dateオブジェクトの機能を使って、さまざまな日時を表示してみましょう。Dateオブジェクトを使うと、「年」「月」「日」などを取り出して、Webページに表示できます。ここでは、2-1節のSample2-1-1.htmlのJavaScriptの部分を変更して、さまざまな日時表示をしていきます。

年を表示する

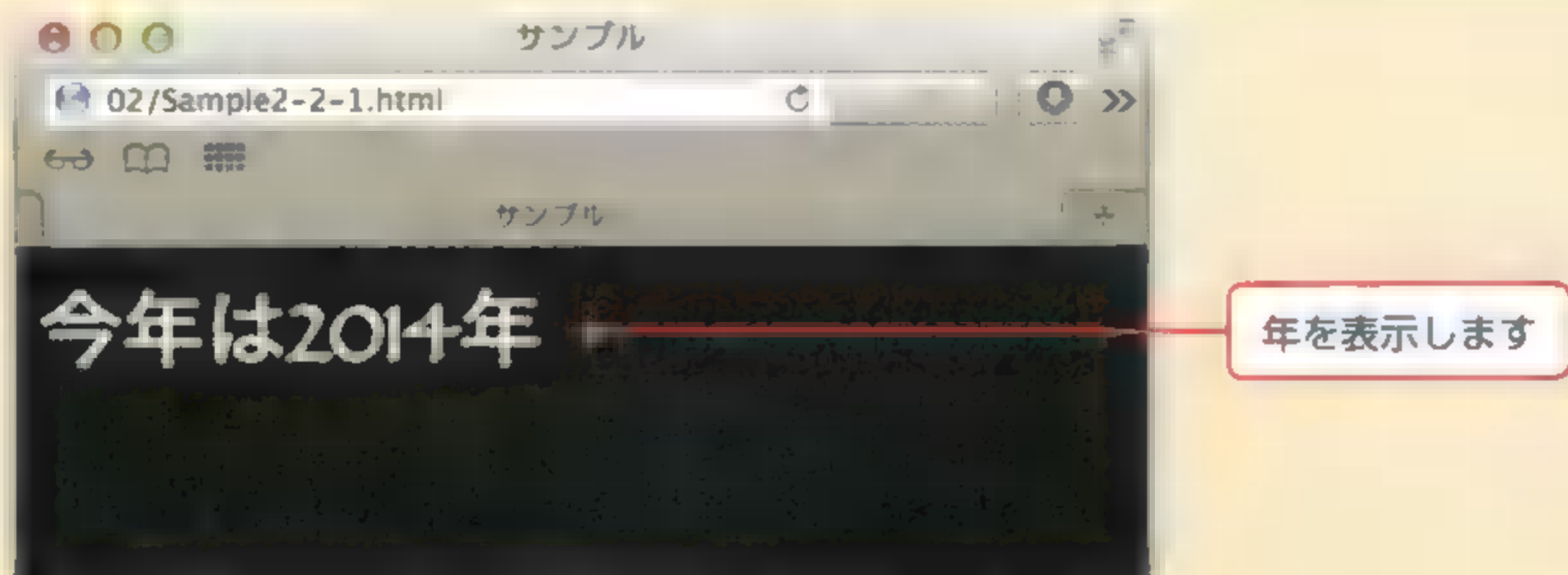
年を調べるにはgetFullYear()メソッドを使います。調べた年に、「今年は」「年」という文字列を付け加えてみました。

Sample2-2-1.html ■ 年を調べる

```
...
<script type="text/javascript">
var d = new Date();
var y = d.getFullYear();
document.writeln("今年は" + y + "年");
</script>
...
```

年を調べます

文字列を付け加えました

■ Sample2-2-1の表示

月・日を表示する

「月」を調べるにはgetMonth()メソッド、「日」を調べるにはgetDate()メソッドを使います。「今日は」「月」「日」の文字列を付け加えています。

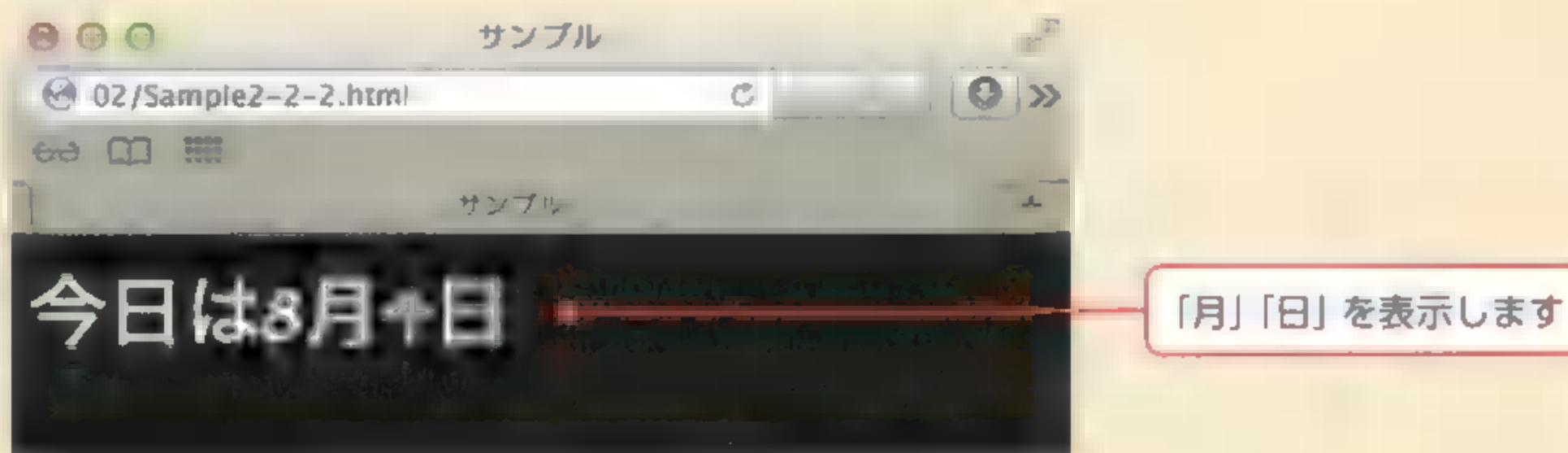
Sample2-2-2.html ■ 月日を調べる

```
...
<script type="text/javascript">
var d = new Date();
var mt = d.getMonth() + 1;
var dt = d.getDate();
document.writeln("今日は" + mt + "月" + dt + "日");
</script>
...
```

「月」を調べます

「日」を調べます

■ Sample2-2-2の表示

Tips

月の値は、0からはじまり11でおわります。つまり、最初の1月が0、2月が1となり、12月は11となります。このため、上のプログラミングでは+1としていることに注意してください。

時刻を表示する

「時」を調べるにはgetHours()メソッド、「分」を調べるにはgetMinutes()メソッドを使います。時刻を調べる方法も同じように使えることがわかるでしょう。

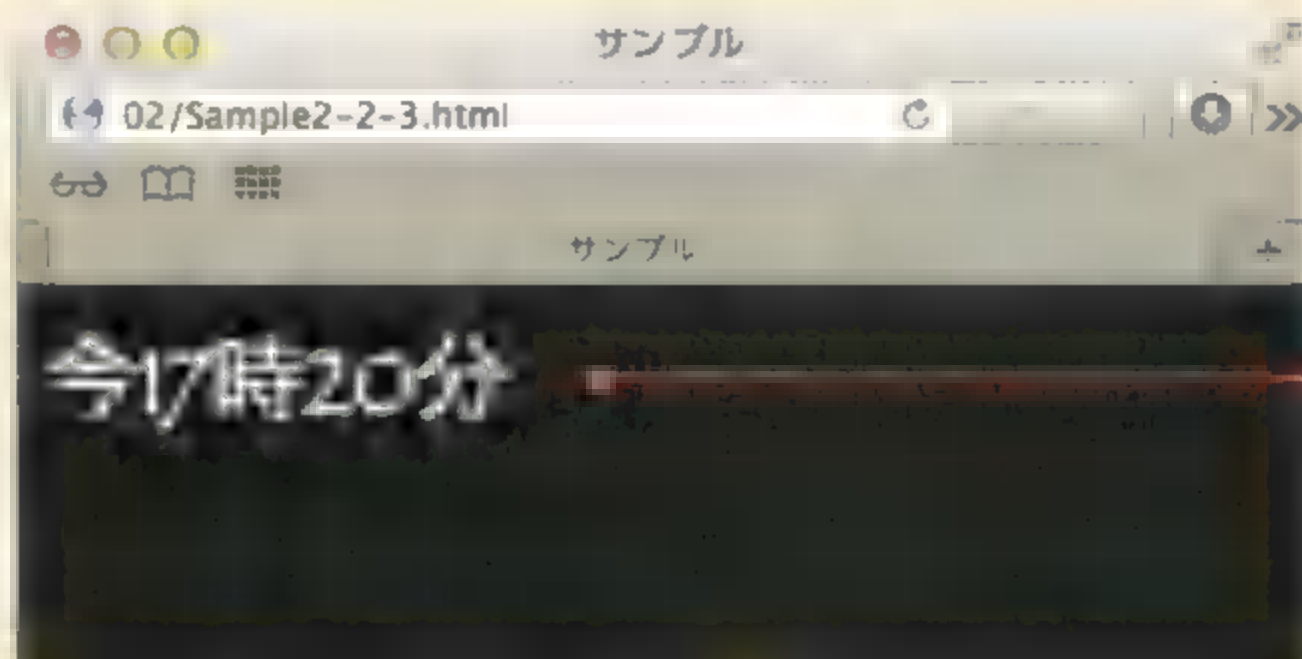
Sample2-2-3.html ■ 時刻を調べる

```
...
<script type="text/javascript">
```

```
var d = new Date();
var h = d.getHours();
var m = d.getMinutes();
document.writeln("今" + h + "時" + m + "分");
</script>
...
```

「時」を調べます

「分」を調べます

■ Sample2-2-3の表示

コメント　Column

プログラムは、コンピュータに理解させるための言語ですから。人間にとっては内容がわかりにくいときがあります。あとから読み返したときに内容がわからなくならないように、プログラムの中に日本語でメモを書いておくことができます。

//のあとに続けてメモを入力します。コンピュータは、//の行をコンピュータへの指示とはみなさないため、メモを入力しておくことができるのです。これを**コメント**といいます。

```
...
<script type="text/javascript">
//日付を表示する
var d = new Date();
...
```

コメントを入力してみました

テクニックを学ぼう

いかがでしょうか。日時を表示することができたでしょうか？　ここで紹介したプログラムでは、少し高度なテクニックを使っています。サンプルで使ったテクニックについて、もう少しく

わしく解説しておきましょう。Sample2-2-3.htmlの次の２行で解説していきます。

```
var m = d.getMinutes();
document.writeln("今" + h + "時" + m + "分");
```

値を記憶する

Dateオブジェクトを利用すれば、日時などを調べることができます。このとき調べた値は、まずコンピュータ内の領域に記憶しておかなければなりません。このためには、前の節でも紹介した「変数」を使います。

変数はvarで名前を準備するのでした。準備したら、＝を使って右辺の値を左辺に設定（記憶）することができます。つまり、次の指定で、変数minに「分」を記憶することができます。「年」「月」「日」「時」など、ほかの情報もこの方法で記憶し、表示しています。

```
var min = d.getMinutes();
```

変数minに「分」を記憶します

＝という記号は、＝の左の変数に右の値を記憶させるために使われます。
「変数 ＝ newオブジェクト( );」の場合は、作成したオブジェクトの場所を記憶させているのです。一般的に使われる「○と×が等しい」という意味とは少し違うので注意してください。なお、一度記憶した後に、もう一度＝を使って設定すると、変数の値は上書きされて変更されます。

文字列は" "で囲う

記憶した値は、Documentオブジェクトのwriteln()メソッドを使ってWebページに表示しています。ここでは、ページ上に表示するときに、「今」や「時」「分」などの文字列を付け加えて表示することにしました。このようにあらかじめ決まっている文字列は、" "または' 'で囲うことになっています。

なお、変数は" "や' 'では囲みませんので注意してください。

文字列を" "で囲みます

```
document.writeln("今" + hour + "時" + min + "分");
```

＋でつなぎます

「"」はダブルクォーテーション、「'」はシングルクォーテーションと呼びます。JavaScriptの文字列は、どちらの記号を使ってあらわしてもかまいません。ただし、"で始まった場合は"で閉じ、'で始まった場合は'で閉じる必要があります。記号が対応していなければならないのです。

変数と文字列を＋でつなぐ

変数と文字列は、＋でつなぎます。つけたす文字列を工夫すれば、さまざまなメッセージを表示することができるでしょう。いろいろなWebページを作成してみてください。

Documentオブジェクト

ここでは、Documentオブジェクトのwriteln()メソッドを使って、Webページに日時を埋め込んできました。

Documentオブジェクトはnewを使わずに、「document」という決まった名前の変数で扱うことができる特別なオブジェクトです。documentはWebページをあらわす特別なオブジェクトとなっています。このため「document.writeln(表示したい文字列);」という指定でWebページにメッセージを埋め込むことができているのです。

Documentオブジェクトのデータと機能は、writeln()メソッドのほかにも、次のものなどがあります。

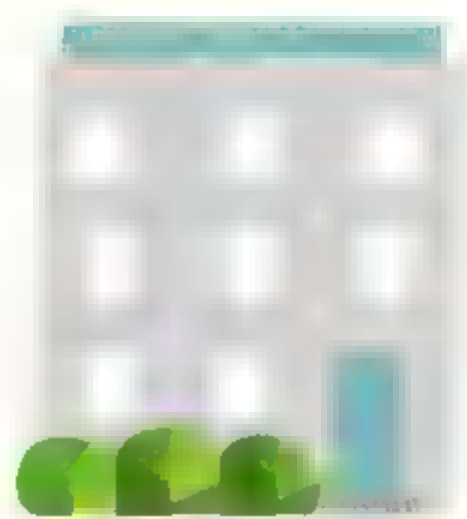
■ Documentオブジェクトのメンバ

メンバ	説明
clear()	文書内容をクリアする
open()	文書出力を開始する
close()	文書出力を終了する
write()	文書に出力する
writeln()	文書に出力・改行する
getSelection()	選択文字列を取得する
cookie	クッキーの値
domain	文書のドメイン
lastModified	文書の最終更新日
referrer	リンク元URL
title	タイトル
location	文書のURL
URL	文書のURL

2-3 状況に応じた処理をしよう

チャレンジ!! 平日・休日を表示しよう

今度は日時を調べて、平日・休日に対応する画像をWebページに表示することを考えてみましょう。平日と休日に分けて、次のような名前の画像を２つ用意してみました。この場合、日時によって違う画像を表示する必要があるでしょう。そのようなページには、どのようにプログラムを作成すればよいでしょうか？

■ 平日の「wday.jpg」

■ 休日の「hday.jpg」

曜日によって違う画像を表示するには？

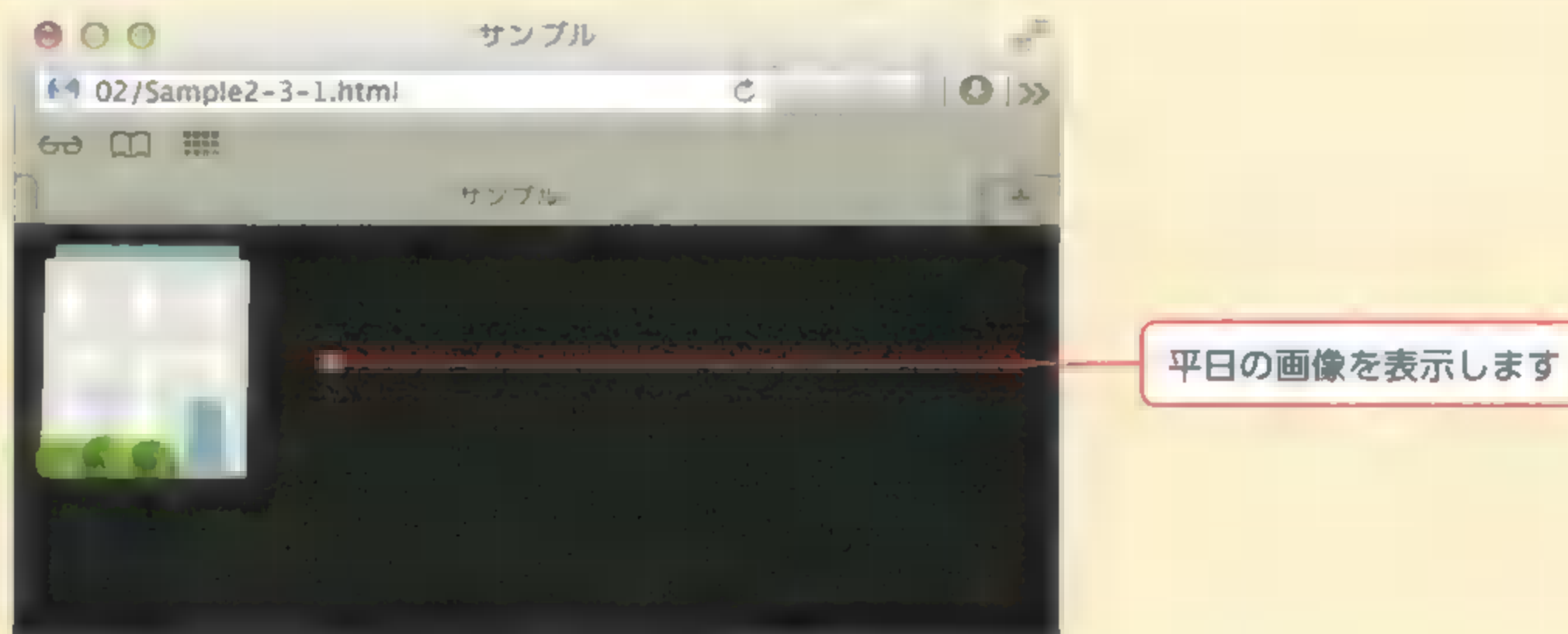
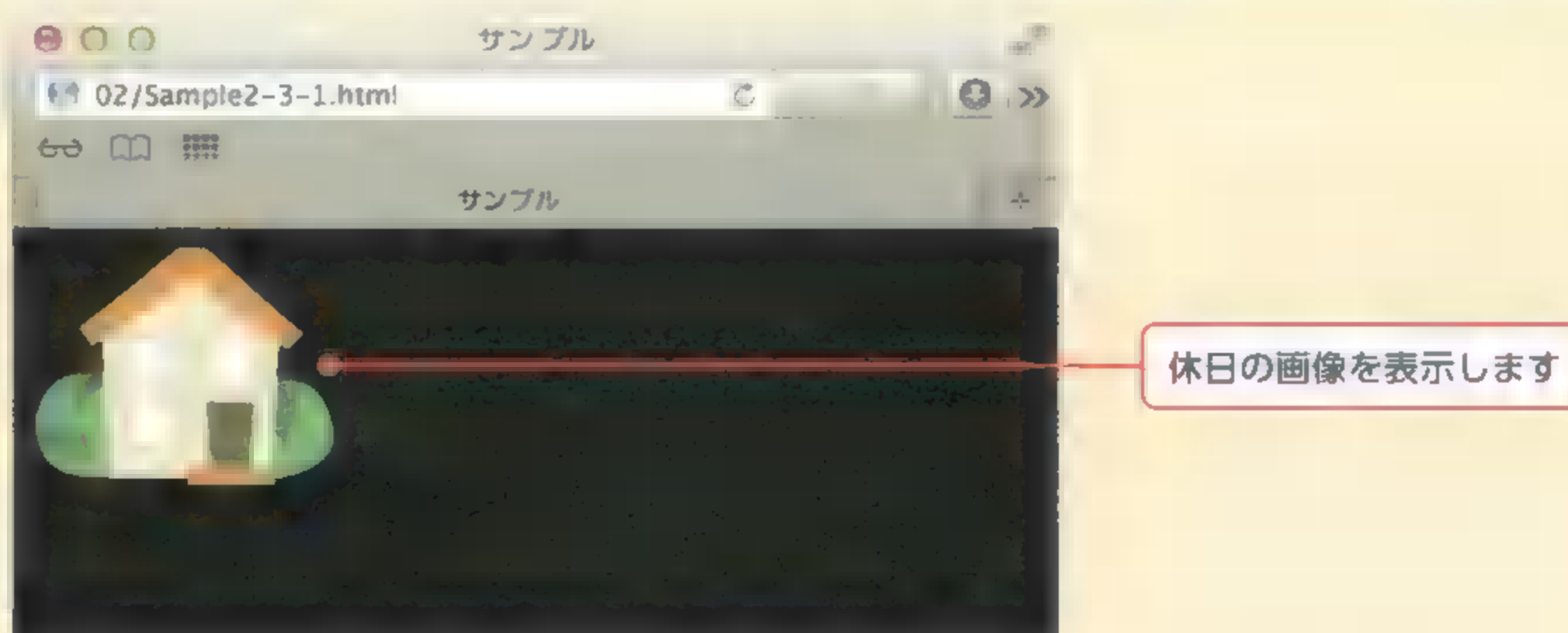
日時に応じて違う画像を表示するには、平日と日曜日によって場合分けをし、それぞれの場合に応じた画像を表示する処理をします。DateオブジェクトのgetDay()メソッドを使うと、曜日を調べることができます。日曜日であれば0、月曜日であれば1というように、0〜6の値として得ることができます。

Sample2-3-1.html ■ 場合分けをする

```
<!DOCTYPE html>
<html lang="ja">
<head>
<link rel="stylesheet" href="Sample.css">
<title>サンプル</title>
</head>
```

```
<body>
<script type="text/javascript">
var d = new Date();
var day = d.getDay();
if(day == 0){
   document.writeln('<img src="hday.jpg">');
}
else{
   document.writeln('<img src="wday.jpg">');
}
</script>
</body>
</html>
```

曜日を調べます
日曜日であれば・・・
休日の画像を表示します
それ以外であれば・・・
平日の画像を表示します

■ Sample2-3-1の表示（平日の場合）

■ Sample2-3-1の表示（日曜日の場合）

場合分けをしよう

ここではJavaScriptを使って、次のような内容の処理を書いています。

```
if(日付が日曜日である){
    休日の画像を表示する
}
else{
    それ以外の画像を表示する
}
```

場合分けをします

もし、日付が日曜日だったら休日の画像を、そうでない場合には平日の画像を表示するようにしているのです。

日曜日の場合は、ifの後の{ }で囲まれた部分の処理が行われます。elseのあとの{ }の処理は行われません。また、日曜日以外の場合は、elseのあとの{ }内の処理が行われ、ifのあとの{ }で囲まれた部分の処理は行われません。曜日によって異なる画像が表示されるように、場合分けをしているのです。

こんなふうに、JavaScriptでは、if～elseという文を使うと状況に応じて違う処理をすることができます。

書式 if~else文

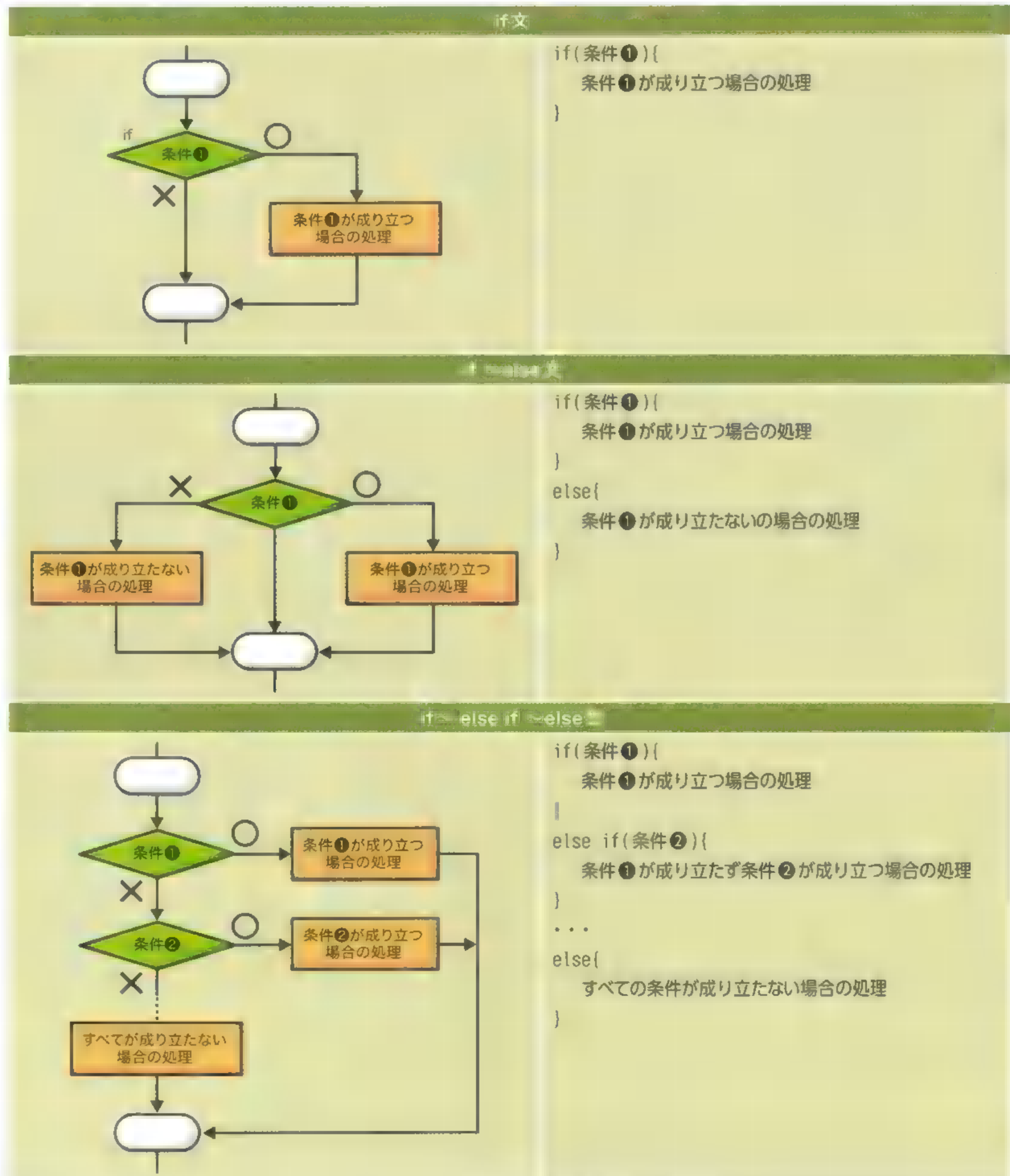
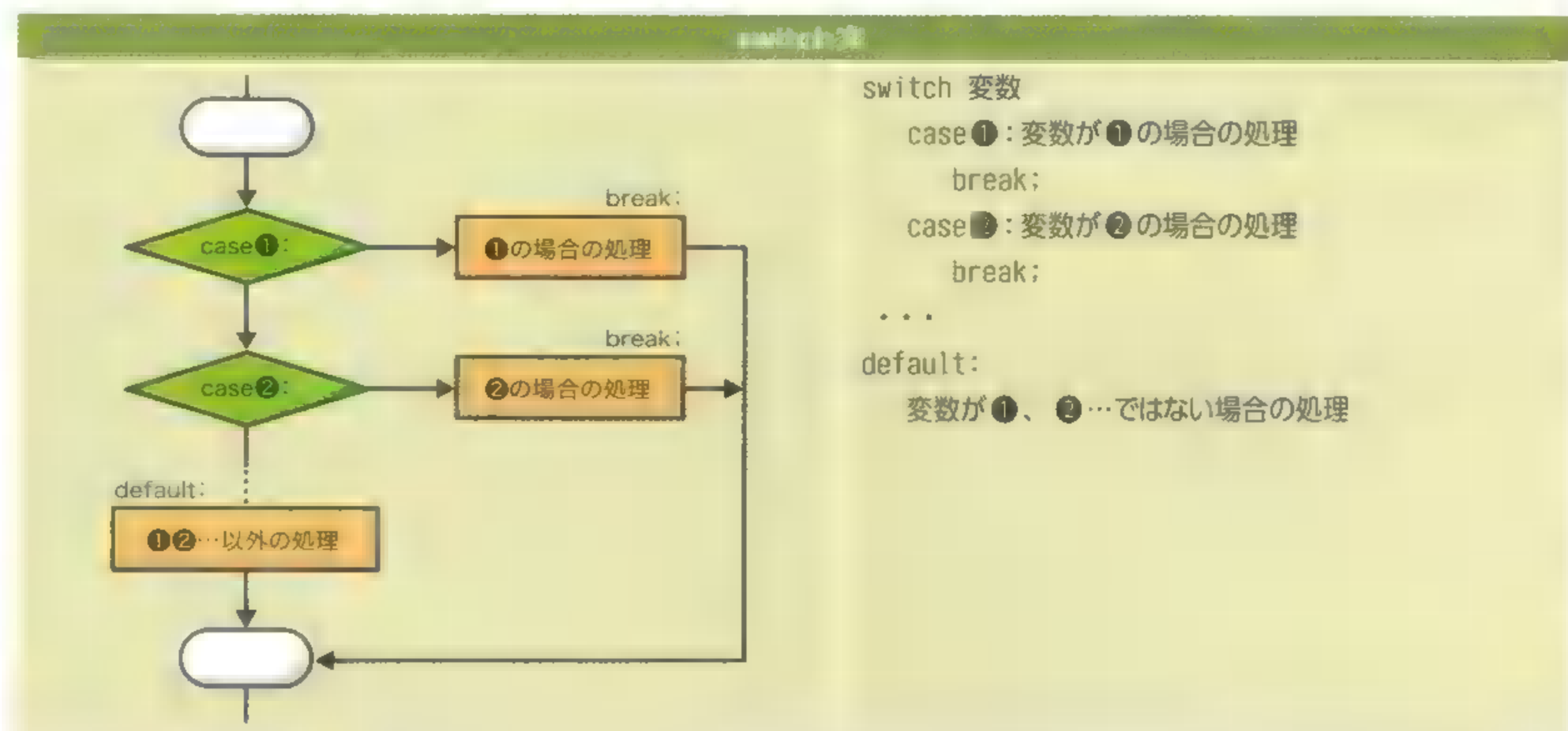
```
if(条件❶){
    条件❶が成り立つ場合の処理をする
}
else{
    それ以外の場合の処理をする
}
```

これで、Webページを開いた曜日に応じて、別の画像を表示する処理ができるようになるのです。

{ }内には複数の文を入力できるので、複雑な処理もできるでしょう。なお、{ }内の処理が1つの文であるときには、{ }を省略することもできます。

場合分けのバリエーション

JavaScriptの場合分けの方法には、いくつかの種類があります。場合分けのバリエーションを紹介しましょう。

■場合分けのバリエーション

応用 日時によって違う処理をする

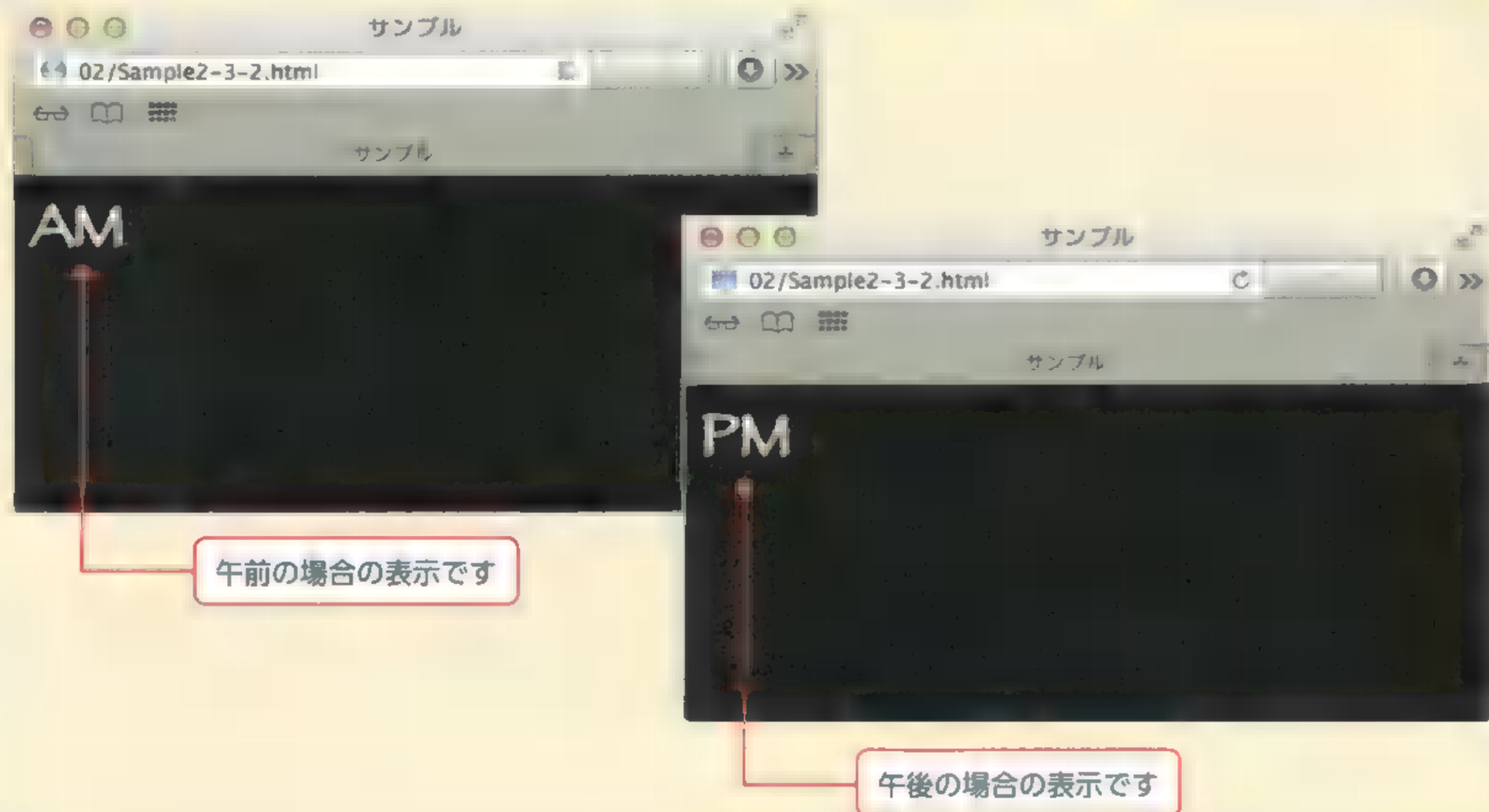
バリエーションに富んだ場合分けを使いこなすと、状況に応じてきめ細かい処理を行ったWebページを表示することができます。そこで、ここで紹介した場合分けの処理を使って、実際にいろいろなプログラムを作成してみましょう。

午前と午後で場合分けする

ユーザーがWebページを開いた時刻が、午前であるか午後であるかによって違う処理をすることにします。ここでは、それぞれ「AM」「PM」と表示を分けてみましょう。

Sample2-3-2.html ■ 午前・午後で場合分けする

```
...
<script type="text/javascript">
var d = new Date();
var h = d.getHours();
if(h <= 12){
   document.writeln("AM");   ← 午前の場合の処理です
}
else{
   document.writeln("PM");   ← 午後の場合の処理です
}
</script>
...
```

■ Sample2-3-2の表示

土日と平日で場合分けする

ユーザーがWebページを開いた時刻が、土日であるか平日であるかによって違う処理をすることにします。switch文を使うと、caseのあとに指定した変数の値によって、異なる処理ができます。変数の値によってcaseからbreak;までの処理が行われるのです。

ここでは、土曜（case 6:）と日曜（case 0:）については同じ処理をしています。

Sample2-3-3.html ■ 平日・休日で場合分けする

```
...
<script type="text/javascript">
var d = new Date();
var day = d.getDay();
switch(day){
   case 0:
   case 6:
      document.writeln('<img src="hday.jpg">');
      break;
   default:
      document.writeln('<img src="wday.jpg">');
}
</script>
...
```

土曜日・日曜日の場合の処理です

そのほか（平日）の処理です

■Sample2-3-3の表示

「条件」を考えよう

ところで、場合分けをするときには、「(もし) ○○だったら・・・」という条件を書くことが必要です。こうした条件はどのように書けばよいのでしょうか？　ここで少しみておくことにしましょう。

「○○である」という条件は、次の記号を使って作ります。

■条件を作る記号

記号	名前
>	より大きい
>=	以上
<	未満
<=	以下
==	等価
!=	非等価

たとえば、「もし12時以降だったら・・・」という条件は、次のように書きます。

```
h >= 12
```

12時以降であれば成り立つ条件です

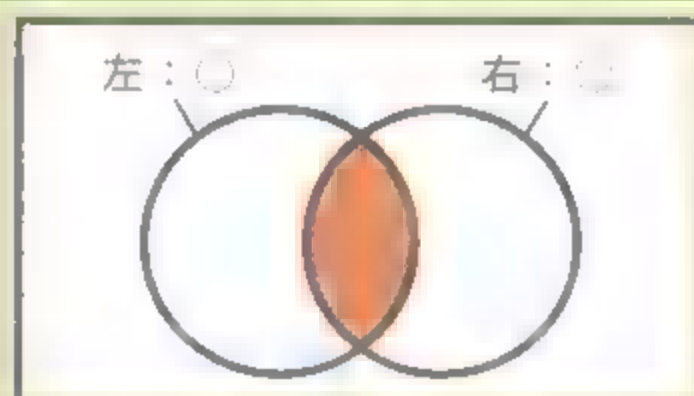
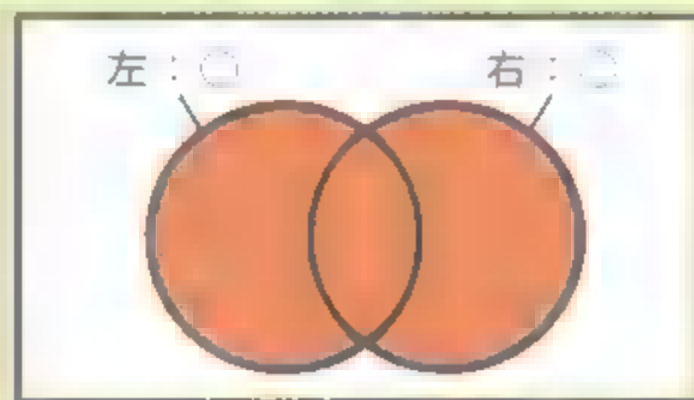
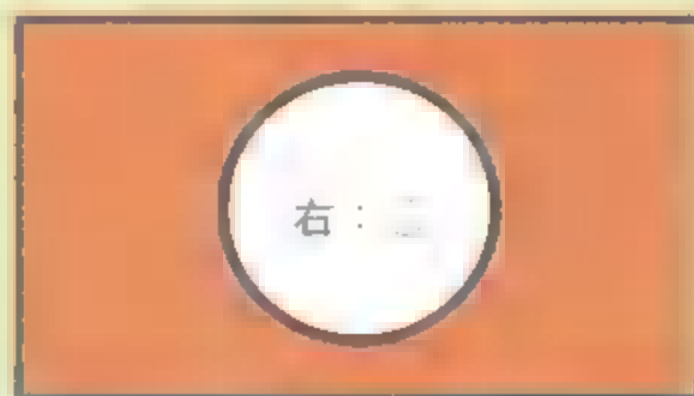
「もし12時でない場合は・・・」という条件は、次のように書くことになります。さまざまな条件を書くことができるでしょう。

```
h != 12
```
12時でなければ成り立つ条件です

表中の==という記号は、記号の左右が等しいことを調べるための記号です。たとえば、hが12であるかを調べる場合には「h==12」とします。=と=の間にスペースなどは入れないので注意してください。
また、「h=12」は、hに12を記憶させるコードですので違いに注意が必要です。

複雑な条件も考えよう

次の記号をあわせて使うと、条件どうしを組み合わせて、さらにきめ細かい場合分けができます。

■複雑な条件を作る記号

この&&や||、!といった記号は、条件を組み合わせるために使います。たとえば、次の条件をみてください。

```
if(h >= 9 && h < 17)
```
9時以降かつ17時以前であれば成り立つ条件です

&&は「・・・でありかつ・・・」という条件をつくる記号です。左右2つの条件がともに成り立つ場合にだけ、条件が成り立つことになります。

ここでは「9時以降であり、かつ17時以前である」という条件となっています。つまり、「9時から17時までである」という条件を書いたわけです。

オープンとクローズの表示分け

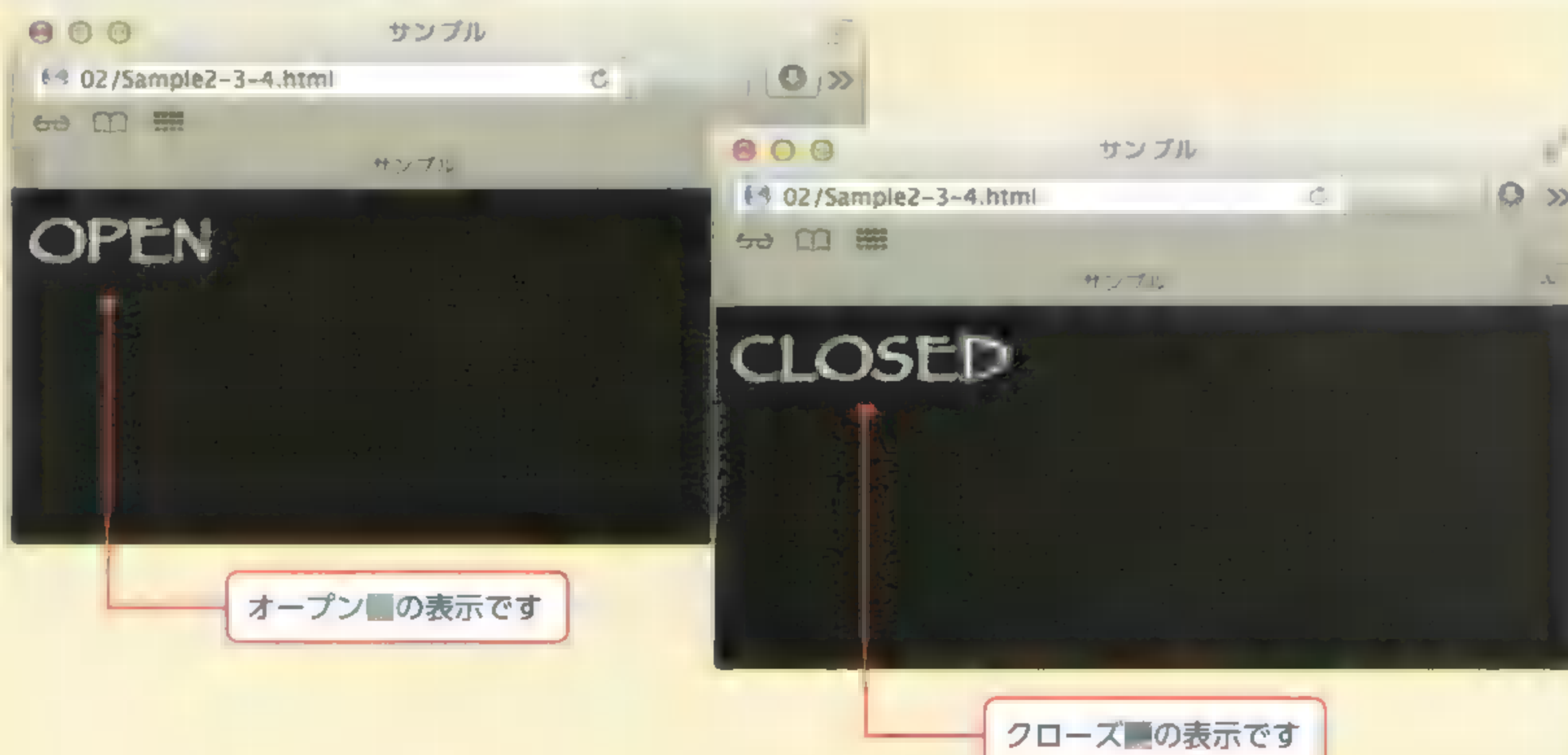
たとえば、9時〜17時に営業している店舗のオープン・クローズを表示するためには、次のようなプログラムとすればよいでしょう。条件が成り立つ場合を確認してみてください。

Sample2-3-4.html ■ 条件を組み合わせて場合分けする

```
...
<script type="text/javascript">
var d = new Date();
var h = d.getHours();
if(h >= 9 && h < 17){
   document.writeln("OPEN");
}
else{
   document.writeln("CLOSED");
}
</script>
...
```

9時〜17時(オープン時)の処理です

クローズ時の処理です

■Sample2-3-4の表示

2-4 繰り返し処理をしよう

チャレンジ! 画像を全部表示しよう

前の節では、状況に応じて異なる画像を表示するようにしました。今度はたくさんの画像の全部をまとめて表示することを考えましょう。たとえば、pic0.jpg～pic5.jpgの6枚の写真があるとします。これらを全部まとめて表示してみましょう。どのようにすればいいでしょうか？

コード中に画像の名前を全部書いて表示することもできます。ですが、JavaScriptの機能を使えばもっと簡単に表示することができます。

■ 写真

pic0.jpg　pic1.jpg　pic2.jpg
pic3.jpg　pic4.jpg　pic5.jpg

繰り返して表示しよう

すべての画像を、まとめて表示するには次のようにします。

Sample2-4-1.html ■ 繰り返し表示する

```
<!DOCTYPE html>
<html lang="ja">
<head>
<link rel="stylesheet" href="Sample.css">
<title>サンプル</title>
</head>
<body>
<script type="text/javascript">
for(var i=0; i<6; i++){
   document.writeln('<img src="pic' + i + '.jpg">');
}
</script>
</body>
</html>
```

画像を表示します

■ Sample2-4-1 の表示

すべての画像が表示されます

繰り返し処理を書いてみよう

ここでは、次のfor文という処理によって6枚の画像を全部表示しています。

❶ iを0と設定し・・・
❷ 6未満である間・・・
❸ 毎繰り返しの最後にiを1つふやします

```
for(var i=0; i<6; i++){
  document.writeln('<img src="pic' + i + '.jpg">');
}
```

この処理を繰り返します

これは変数iを0と設定し（❶）、6未満である間（❷）、画像を表示する「img」タグを繰り返し作成し続ける処理です。

i++の「++」は、インクリメントと呼ばれ、値を1つずつ増やす処理です（❸）。1回分の繰り返し処理が終わった後に変数の値を1つ増やすので、6回繰り返したときに変数が6となり、繰り返しが終了するのです。

つまり、ここでは次のようにfor文を使って繰り返し処理を行っていることになります。

構文

```
for(var 変数=初期の値; 繰り返しを続ける条件; 繰り返しの最後に行う処理){
  繰り返す処理
}
```

なお、繰り返しの処理の中では、次のように変数iの値を使っています。

変数の値を埋め込んでいます

```
document.writeln('<img src="pic' + i + '.jpg">');
```

繰り返しごとに変数iが1つずつふやされ、pic0.jpg、pic1.jpg…という名前となりますから、ここでは次のようなWebページを作成しているのと同じことになります。

```
<img src="pic0.jpg">
<img src="pic1.jpg">
<img src="pic2.jpg">
...
```

このようなWebページを作成していることになります

つまり、6つの画像がすべてWebページ上に表示されるわけです。

ここではJavaScriptの文字列を指定するために'（シングルクォーテーション）を使い、HTMLで画像名を指定するために"（ダブルクォーテーション）を使いました。'と"は逆にしてもかまいません。ただし、それぞれがきちんと対応されているようにします。また、変数の部分は'や"で囲まずに、文字列を連結する+でつなぐことに注意してください。

インクリメント・デクリメント

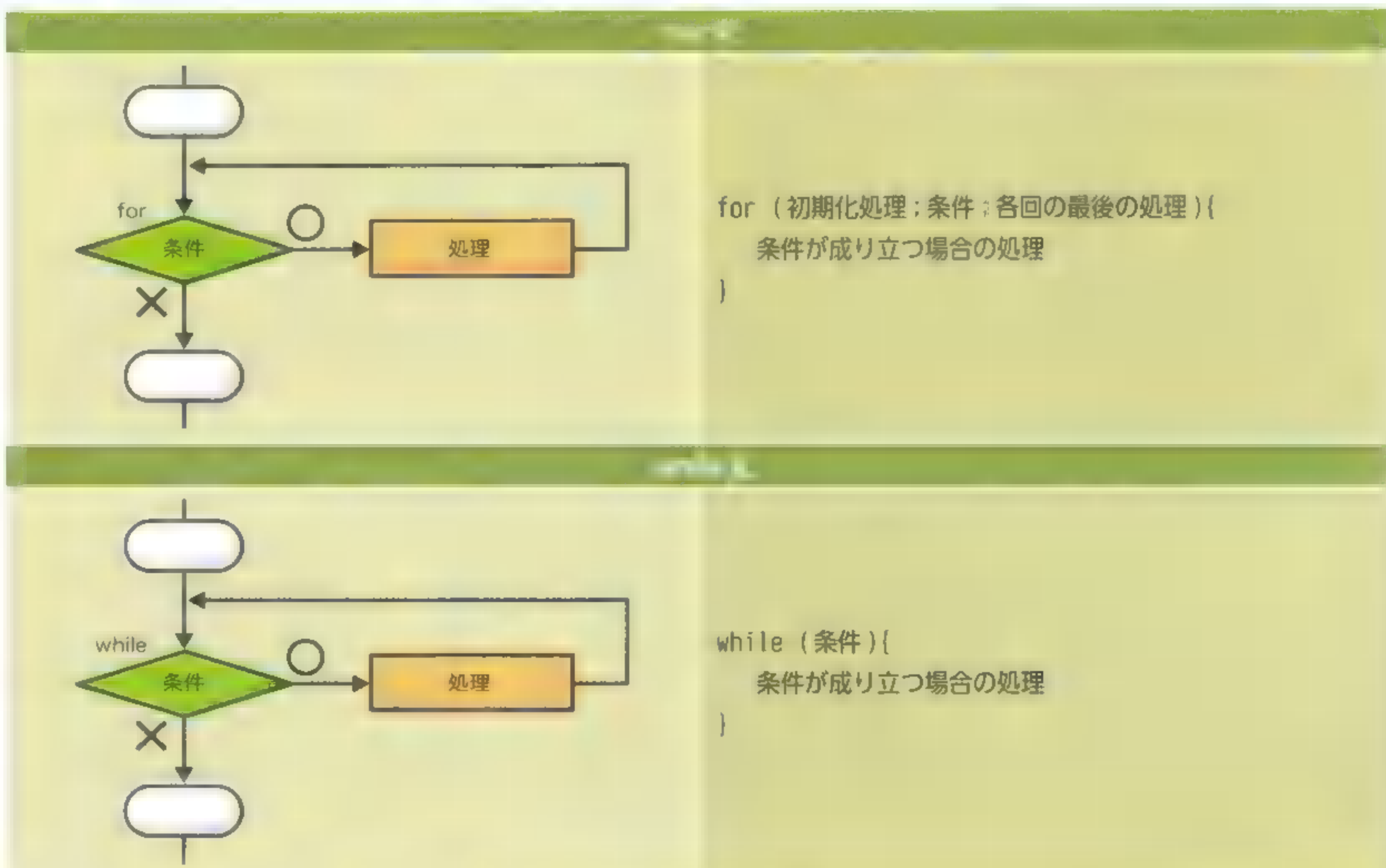
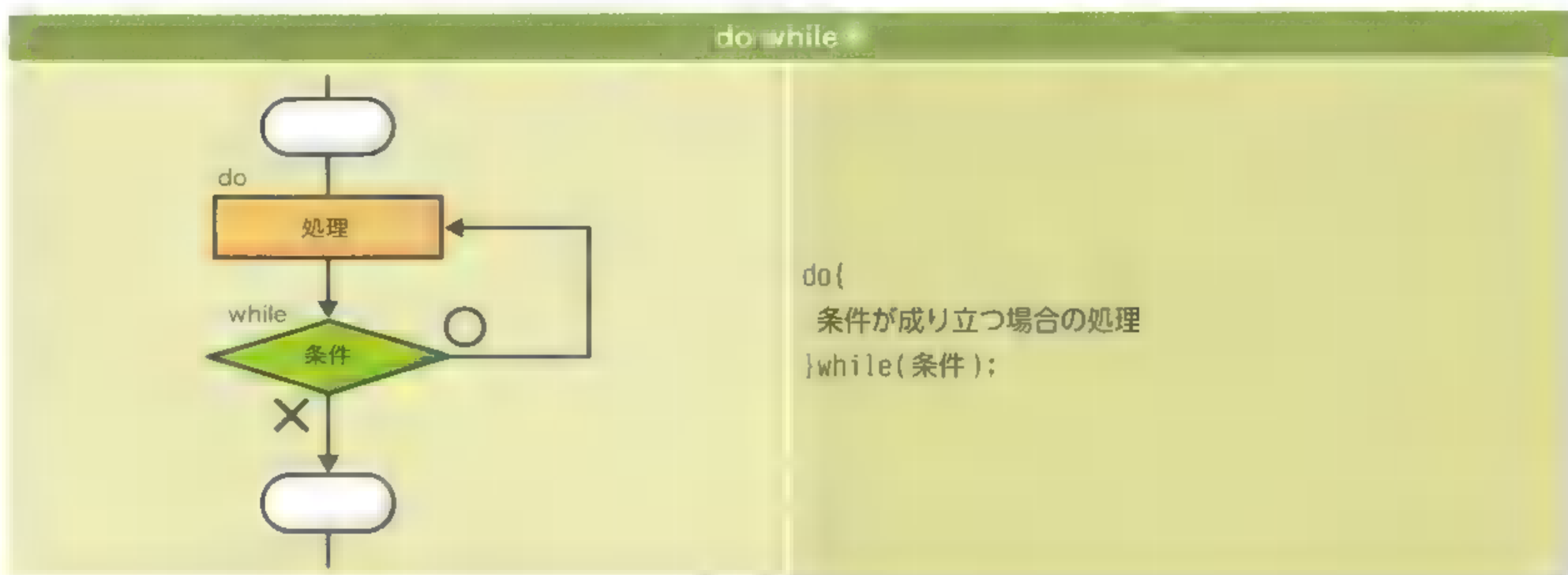
for文の中で使っている++記号は、インクリメントと呼ばれ、値を1つ増やす指定となっています。値を1つ減らす--記号もあり、デクリメントと呼ばれます。

インクリメント・デクリメントは、1つずつ値を増やしたり減らしたりするため、処理の回数を1回ずつカウントするときなどによく用いられます。for文では、インクリメントとデクリメントがよく使われます。

繰り返しのバリエーション

JavaScriptの繰り返し処理には、次のようなバリエーションがあります。

■繰り返し

Tips

繰り返し文はどれも繰り返し処理を行うことができますが、一般的に回数が明確な繰り返しはfor文を使い、そうでない場合にはwhile文を使う機会が多くなっています。

繰り返して処理しよう

繰り返し文を使ってさまざまなWebページを作成することができます。実際にプログラムを作成して確認してみましょう。

写真のリストを作成する

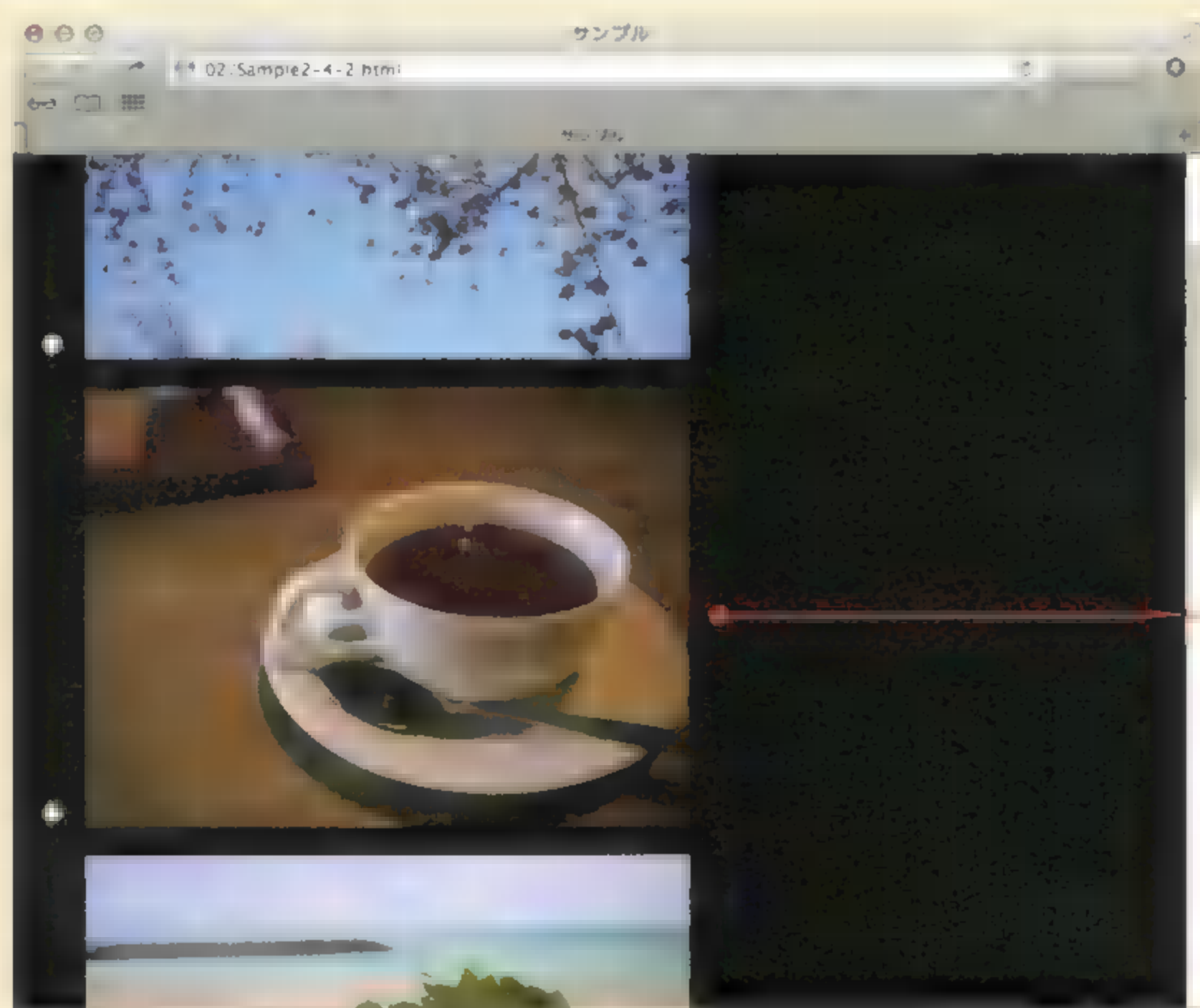
写真のリストを作成してみます。項目をあらわすHTML文書の「li」要素を繰り返し出力します。リスト全体をあらわす「ul」要素は、繰り返しの外で出力します。

Sample2-4-2.html ■ リストを表示する

```
<!DOCTYPE html>
<html lang="ja">
<head>
<link rel="stylesheet" href="Sample.css">
<title>サンプル</title>
</head>
<body>
<script type="text/javascript">
document.writeln('<ul>');          ← 繰り返しの外に書きます
for(var i=0; i<6; i++){

  document.writeln('<li>');        ← リスト項目を繰り返します
```

```
    document.writeln('<img src="pic' + i + '.jpg">');
    document.writeln('</li>');
}
document.writeln('</ul>');
</script>
</body>
</html>
```

繰り返しの外に書きます

■Sample2-4-2の表示

1行の表を作成する

表を作成することもできます。セルをあらわす「td」要素を繰り返すことにします。

表全体をあらわす「table」要素は、繰り返しの外で処理します。この表は1行であるので、行をあらわす「tr」要素も繰り返しの外で処理することにします。

Sample2-4-3.html ■ 表を作成する

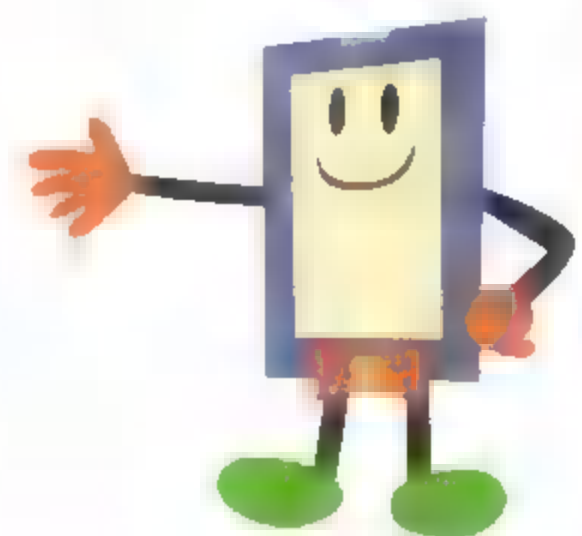
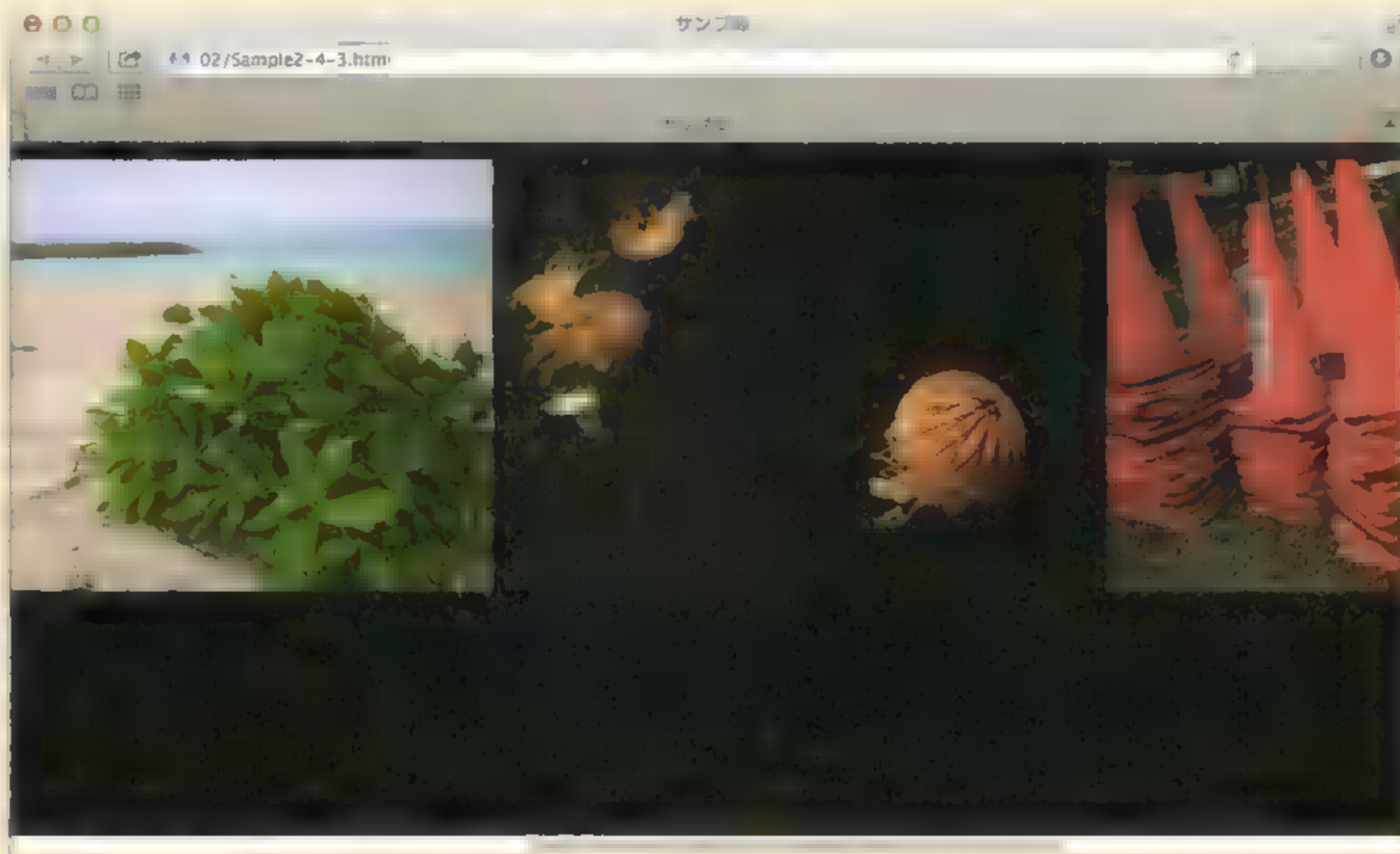
```
<!DOCTYPE html>
<html lang="ja">
<head>
<link rel="stylesheet" href="Sample.css">
<title>サンプル</title>
</head>
<body>
```

```
<script type="text/javascript">
document.writeln('<table>');
document.writeln('<tr>');

for(var i=0; i<6; i++){
  document.writeln('<td>');
  document.writeln('<img src="pic' + i + '.jpg">');
  document.writeln('</td>');
}
document.writeln('</tr>');
document.writeln('</table>');
</script>
</body>
</html>
```

繰り返しの外に書きます

繰り返しの外に書きます

セル項目を繰り返します

■ Sample2-4-3の表示

リストと表　Column

リスト・表をあらわすHTML要素には、次の種類があります。リスト・表は複数の要素から構成されています。

■リスト

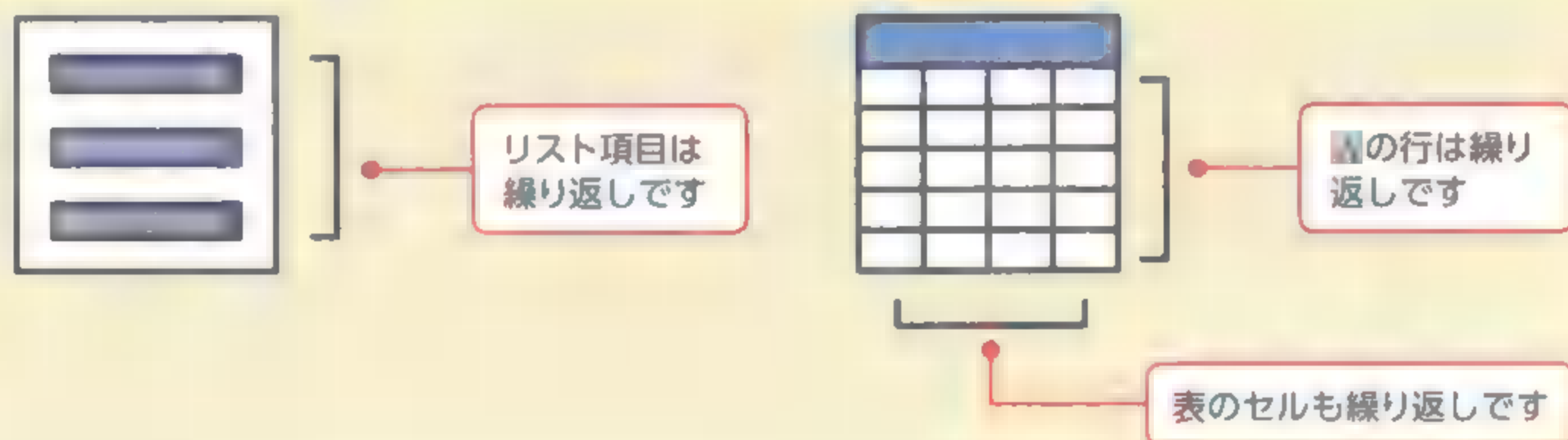
要素	説明
ul	箇条書きリスト
ol	番号つきリスト
li	リスト項目

■表

要素	説明
table	表（テーブル）
caption	表説明
tr	1行
th	表頭
td	表項目

繰り返し文を使いこなそう

繰り返し処理になれることができたでしょうか？　繰り返しを使うときには、繰り返されている部分をみつけてコードを作成していくことが大切です。表やリストは次のようになっていますから、各項目を繰り返し文で書くことができるわけです。

■リストと表

繰り返し文を使うときには、繰り返しの部分と繰り返しでない部分に注意することが必要です。繰り返しでない部分は、繰り返し処理の外側に書いていることに注意してください。

このように、Webページも繰り返し文を使うと簡単に表示できる場合があります。

繰り返しの中の繰り返し

Column

繰り返しの中でさらに繰り返すこともできます。表を２行以上繰り返しする場合は、次のように繰り返すことになります。

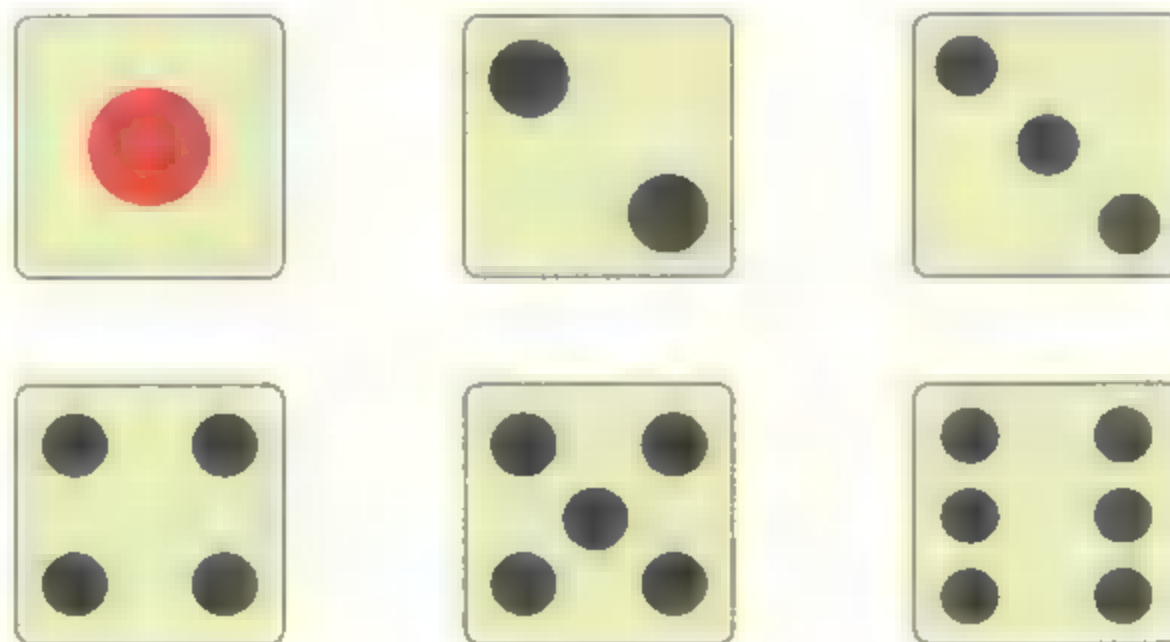
```
for(var j=1; j<=6; j++){
   document.writeln('<tr>');      ← 行項目を繰り返します

   for(var i=1; i<=6; i++){
      document.writeln('<td>');   ← 行内のセル項目を繰り返します
   }
}
```

2-5 きめ細かく処理しよう

チャレンジ!! 組み合わせて処理しよう

場合分けと繰り返し処理を組み合わせれば、さらにきめ細かい処理をすることができます。

たとえば、次のようなサイコロの目の画像を用意してみましょう。このサイコロの目のうち、条件に合致したものだけを表示してみます。ここでは、偶数の目だけを表示してみることにしましょう。

■ サイコロの目（dice1.jpg～dice6.jpg）

偶数だけ表示しよう

偶数だけ表示するには、次のようにif文と繰り返し文を組み合わせるとかんたんです。

Sample2-5-1.html ■ 偶数を表示する

```
<!DOCTYPE html>
<html lang="ja">
<head>
<link rel="stylesheet" href="Sample.css">
<title>サンプル</title>
</head>
<body>
```

```
<script type="text/javascript">
for(var i=1; i<=6; i++){
    if(i % 2 == 0){
        document.writeln('<img src="dice' + i + '.jpg">');
    }
}
</script>
</body>
</html>
```

偶数である場合に・・・

その目の画像を表示します

■ Sample2-5-1の表示

計算をするには？

なお、ここでは偶数を調べるために、次のような条件を作っています。

```
if(i % 2 == 0)
```

あまりが0であるかを調べます

A%Bは、AをBでわったときのあまりの数を調べる記号です。ここでは、変数iを2で割ったときにあまりが0となる場合、すなわちiが偶数である場合に処理を行っています。

%のような記号は演算子と呼ばれ、計算を行う記号です。演算子には次の表のような種類があります。%(剰余)を使う計算はあまりなじみがないかもしれませんが、「たしざん」(加算)、「ひきざん」(減算)、「かけざん」(乗算)、「わりざん」(除算)のような計算はよく使うものでしょう。

■演算子

[illegible]	[illegible]
+	加算
-	減算
*	乗算
/	除算
%	剰余
+	単項+
-	単項−

[illegible]	[illegible]
~	補数
=	代入
<<	左シフト
>>	右シフト
++	インクリメント
--	デクリメント
typeof	型

応用 処理を組み合わせていく

このように、場合分けと繰り返し処理を組み合わせると、いろいろなプログラムに応用することができます。もっとたくさんのプログラムを作成して確かめてみましょう。

奇数の目を表示する

今度は偶数ではなく、奇数の目を表示してみましょう。i%2の値が1であるかどうかを調べることにします。ある値を2で割ったときのあまりが1であれば、その数は奇数であるからです。

Sample2-5-2.html ■ 奇数を表示する

```
...
<script type="text/javascript">
for(var i=1; i<=6; i++){
   if(i % 2 == 1){
      document.writeln('<img src="dice' + i + '.jpg">');
   }
}
</script>
...
```

奇数である場合に・・・

その目の[illegible]を表示します

■Sample2-5-2の表示

表を縞模様にする

次に表の列を縞模様にしてみましょう。偶数列だけ色を変更するようにします。

Sample2-5-3.html ■ 縞模様にする

```
...
<script type="text/javascript">
document.writeln('<table>');
document.writeln('<tr>');
for(var i=0; i<6; i++){
    if(i % 2 == 0){                              ← 偶数列である場合に・・・
        document.writeln('<td id="on">');        ← 色を変更します
    }
    else{
        document.writeln('<td id="off">');
    }
    document.writeln('No.'+ i);
    document.writeln('</td>');
}
document.writeln('</tr>');
document.writeln('<table>');
</script>
...
```

■ Sample2-5-3の表示

ここでは、JavaScriptで、id="on"またはid="off"というID属性を指定した2種類の「td」要素を作成しています。ID属性は、HTML文書の要素を特定するために使われるものです。スタイルシートの中で、それぞれの要素に対して異なる色（レイアウト）を指定することで、要素の

色を変更しているのです。色を変更する場合には、こうしたテクニックが使われます。

この章で使っているスタイルシート（Sample.css）の中身を確認しておきましょう。なお#はID名であることをあらわします。

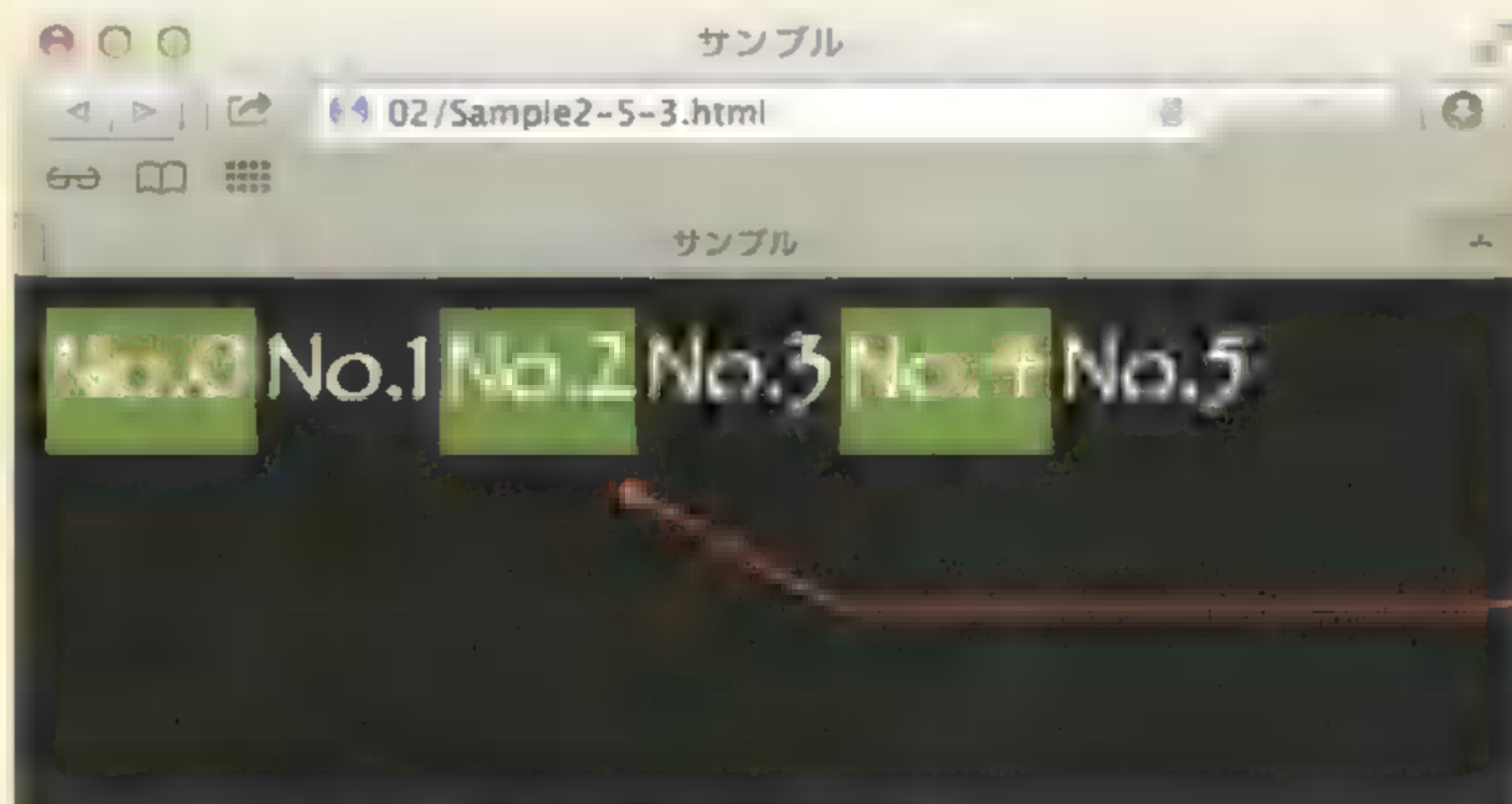
```
...
#on{
   background-color: #2e8b57;
   color: #FFFFF0;
   font-family: fantasy;
}
#off{
   background-color: #000000;
   color: #FFFFF0;
   font-family: fantasy;
}
```

- #on{ — 色つきのセルに適用されるセレクタです
- #off{ — 色なしのセルに適用されるセレクタです

格子模様に変更する

今度は、表の縞模様を変更してみます。表の行と列の縞模様を工夫して、格子模様にしてみます。複雑な模様も、処理を組み合わせることで作成できるのです。

Sample2-5-4.html ■ 格子模様にする

```
...
<script type="text/javascript">
document.writeln('<table>');
for(var i=1; i<=9; i++){
   document.writeln('<tr>');

   for(var j=1; j<=9; j++){

      if((i%2 == 0 && j%2 == 1)||(i%2 == 1 && j%2 == 0)){
         document.writeln('<td id="on">' + (i*j) +'</td>');
      }
      else{
         document.writeln('<td id="off">' + (i*j) +'</td>');
      }
   }
   document.writeln('</tr>');
}
document.writeln('</table>');
</script>
...
```

- if((i%2 == 0 && j%2 == 1)||(i%2 == 1 && j%2 == 0)){ — 「偶数行・奇数列」と「奇数行・偶数列」の色を・・・
- document.writeln('<td id="on">' + (i*j) +'</td>'); — 変更します

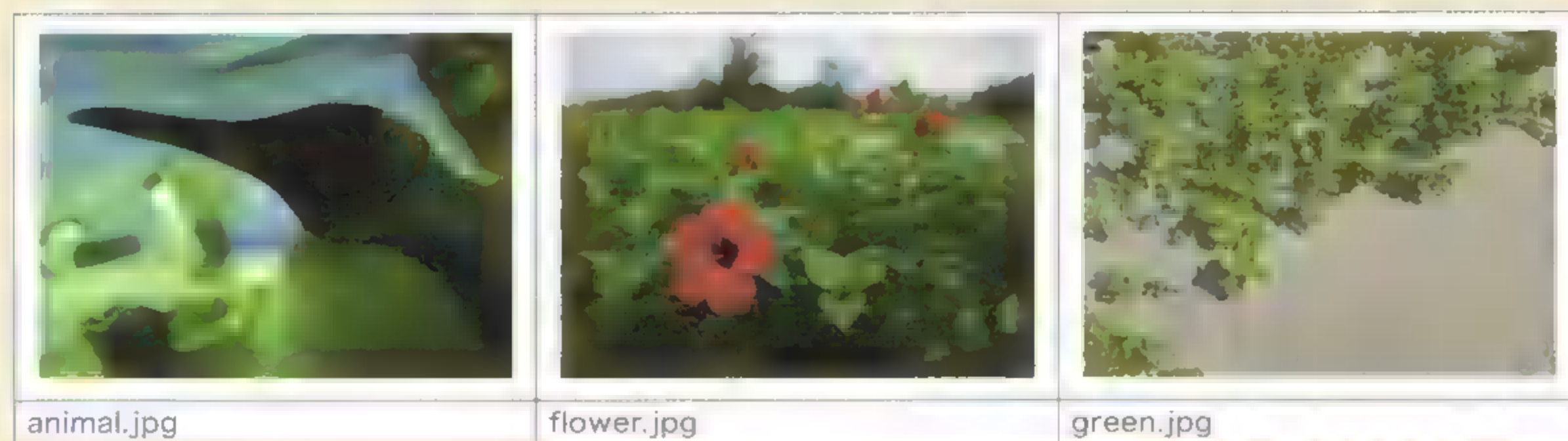
■Sample2-5-4の表示

2-6 データを準備しよう

チャレンジ! データを準備してみよう

これまでの節では、番号をつけて画像を準備してみました。しかし、実際にWebページを作成する場合には、画像名に番号が使われているとは限りません。さまざまな名前の画像データを扱うことになるでしょう。そこで、名前の異なる画像データを扱う方法について考えてみましょう。

■名前の異なる画像データ

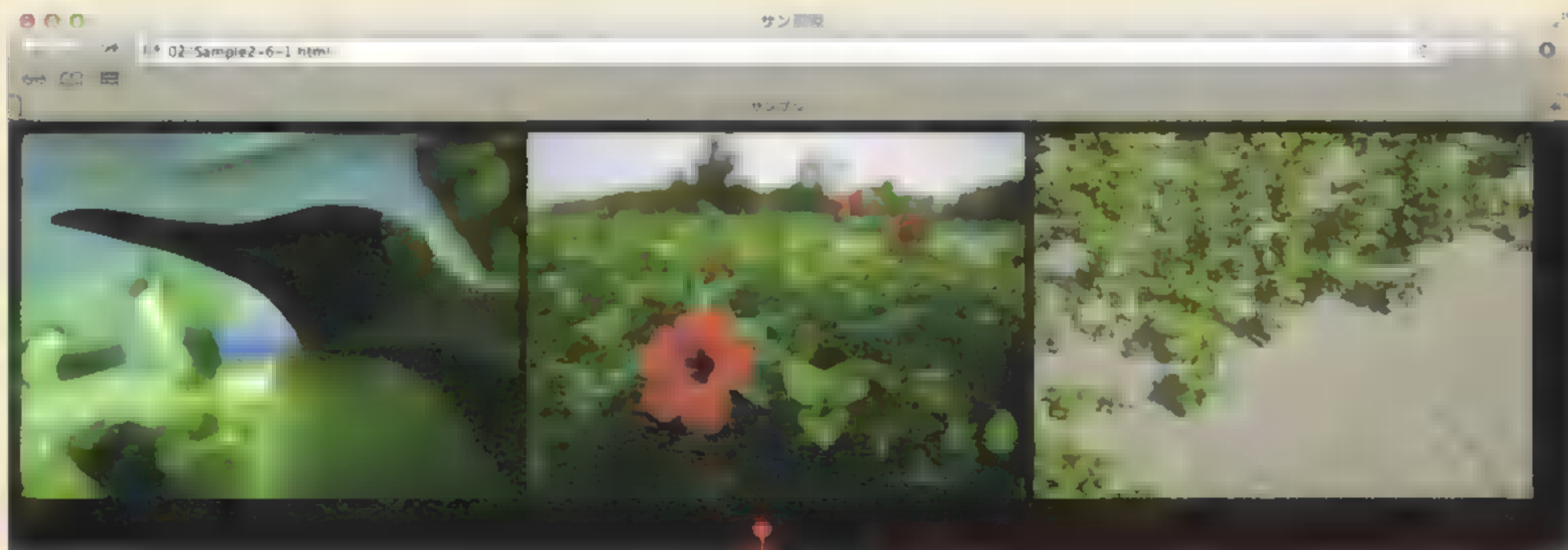
animal.jpg	flower.jpg	green.jpg

画像名データを用意する

さまざまな名前の画像データを扱うとき、プログラムの最初の部分に、画像名のデータを準備しておくと役に立ちます。画像名データを用意できたら、表示してみましょう。

Sample2-6-1.html ■ データを用意する

```
<!DOCTYPE html>
<html lang="ja">
<head>
<link rel="stylesheet" href="Sample.css">
<title>サンプル</title>
</head>
<body>
<script type="text/javascript">
```

```
var str = ["animal.jpg", "flower.jpg", "green.jpg"];   ←❶画像名データの配列を準備します
for(var i=0; i<str.length; i++){
  document.writeln('<img src="' + str[i] + '">');   ←❷配列を表示します
}
</script>
</body>
</html>
```

■ Sample2-6-1の表示

データを準備するには？

Sample2-6-1.htmlの❶の部分が、画像名データを用意しているコードです。

```
var str = ["animal.jpg", "flower.jpg", "green.jpg"];   ←❶画像名データの配列を準備します
```

データの羅列を扱うしくみを、配列といいます。配列は次のように準備します。

構文

```
var 変数 =[データ1, データ2, ・・・];   ←配列の[illegible]方法です
```

上では、strという変数によって画像データの名前を記憶した配列を扱うように準備しているのです。

さて、データを準備したら、配列を利用することができます。❷の部分がデータを利用しているコードです。変数strの[ ]に指定した番号で、どの画像かを指定しています。配列として準

備しておいた画像名を、番号で指定できるわけです。

```
document.writeln('<img src="' + str[i] + '">');
```
❷配列を利用します

つまり配列は次のようにして利用することになります。配列の番号を添字といいます。

構文

変数[添字] — 配列の利用方法です

Tips

配列の添字は0からはじまります。つまり、str[0]がanimal.jpg、str[1]がflower.jpg、str[2]がgreen.jpgをあらわすわけです。3つの要素からなる配列の場合、最後の要素を指定するための添字は、要素数より1つ小さいことに注意してください。str[3]というデータは存在しないことに気をつける必要があります。

■配列データ

応用 配列データを用意して活用する

配列を使って、いろいろなデータを用意し、プログラムに活用してみましょう。

商品名を表示する

商品名データを用意し、すべて表示してみます。「オレンジ」「みかん」「いちご」・・・という商品名データを用意します。あらかじめ商品名データを準備しておき、プログラムの中で活用できれば便利でしょう。

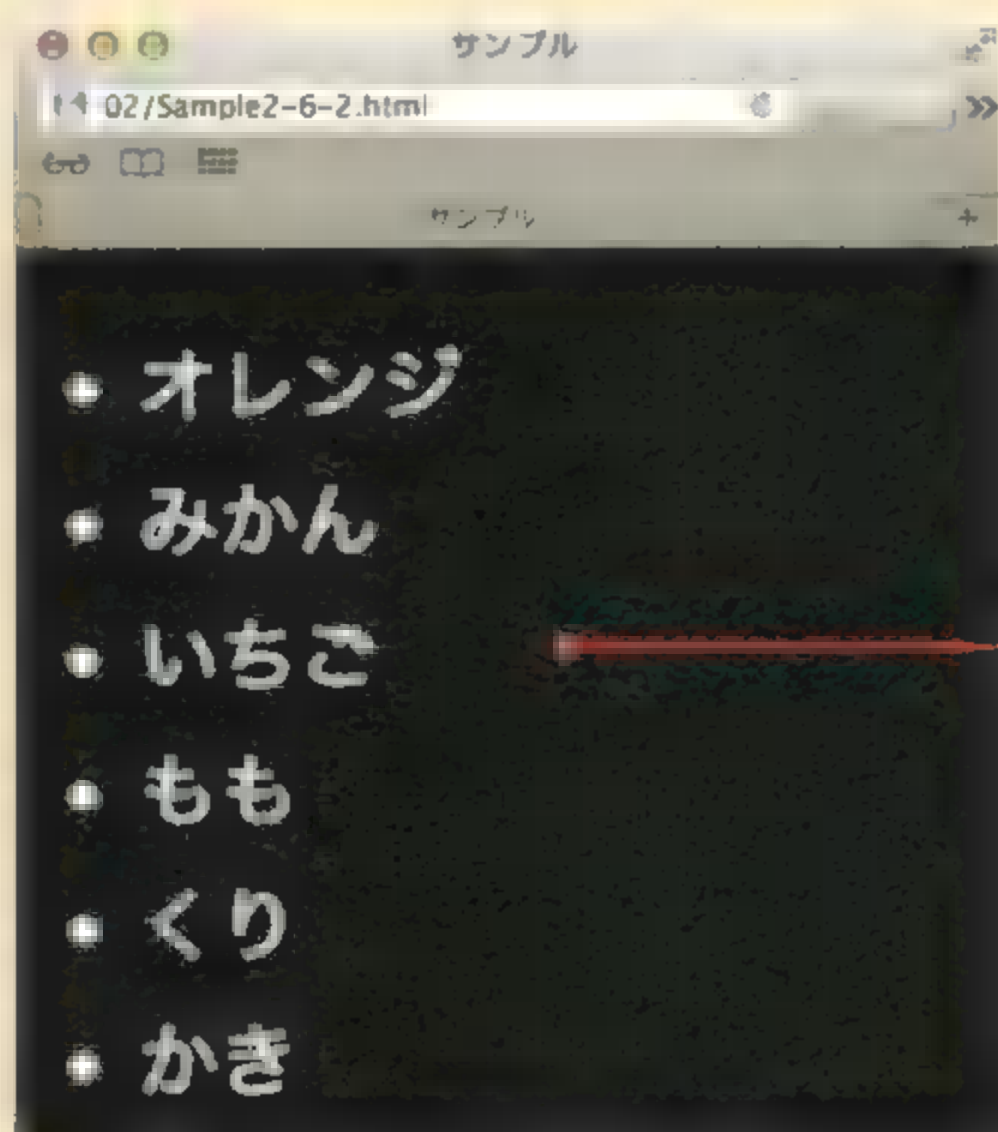
Sample2-6-2.html ■ 商品名を表示する

```
<!DOCTYPE html>
<html lang="ja">
<head>
<link rel="stylesheet" href="Sample.css">
<title>サンプル</title>
</head>
<body>
<script type="text/javascript">
var str = ["オレンジ","みかん","いちご","もも","くり","かき"];
document.writeln('<ul>');
for(var i=0; i<str.length; i++){
   document.writeln('<li>');
   document.writeln(str[i]);
   document.writeln('</li>');
}
document.writeln('</ul>');
</script>
</body>
</html>
```

商品名を配列として準備します

商品名を表示します

■Sample2-6-2の表示

Tips

配列名に「.length」をつけることで、配列中のデータ個数を知ることができます。このコードの繰り返し文中では、「str.length」と指定して、データの個数だけ繰り返しを行っています。

曜日名を表示する

次に、曜日名データを用意し、表示してみます。DateオブジェクトのgetDay()メソッドは曜日を調べることができますが、曜日名ではなく曜日をあらわす数値しか得ることができません。そこで、曜日名を配列データで準備して表示しようというのです。「Sun」「Mon」「Tue」・・・という曜日データを用意します。なお、土曜日（Sat）と日曜日（Sun）の色を変更してみました。

Sample2-6-3.html ■ 曜日名を表示する

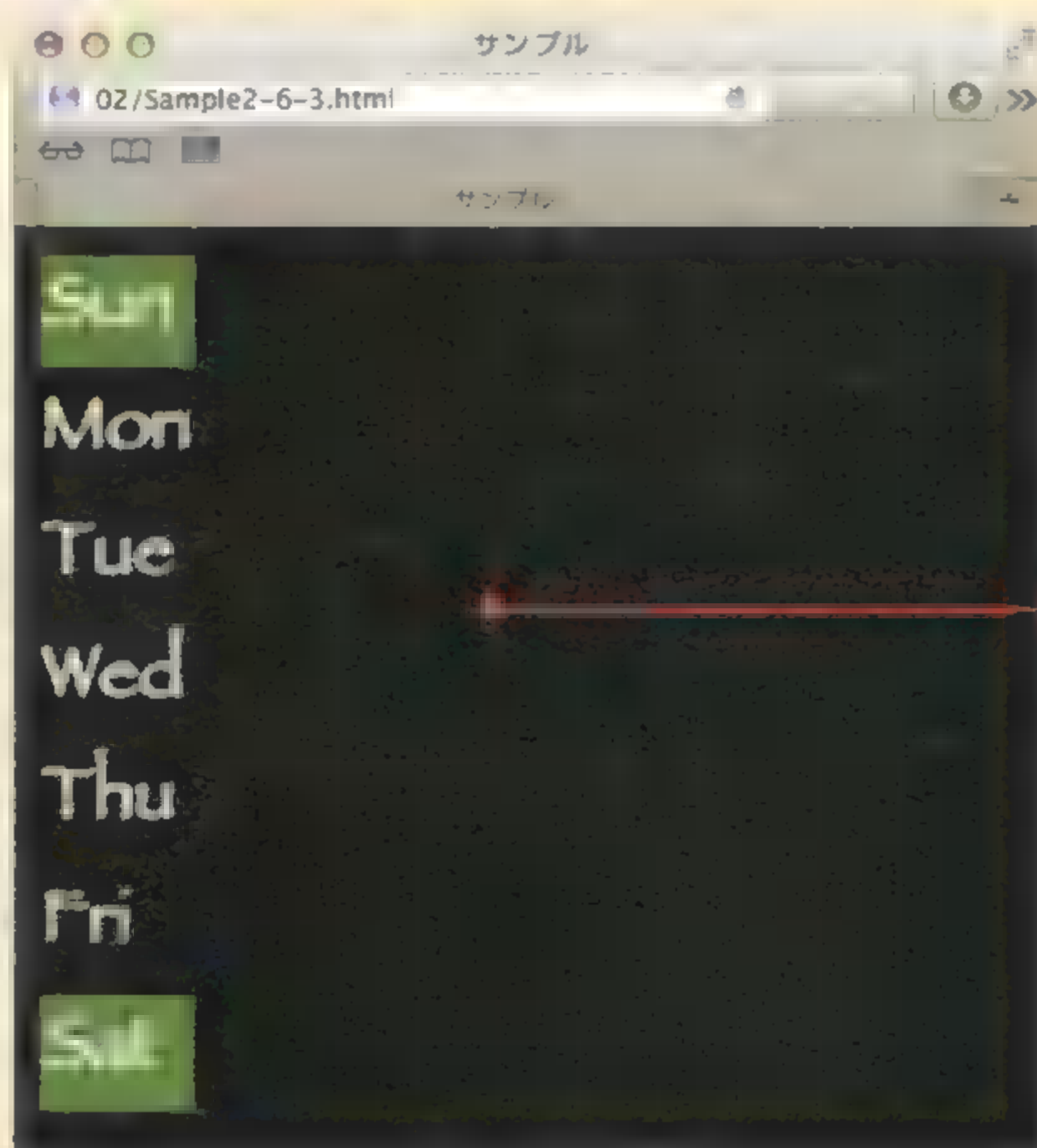
```
<!DOCTYPE html>
<html lang="ja">
<head>
<link rel="stylesheet" href="Sample.css">
<title>サンプル</title>
</head>
<body>
<script type="text/javascript">
document.writeln('<table>');
var str = ["Sun","Mon","Tue","Wed","Thu","Fri","Sat"];
for(var i=0; i<7; i++){
   document.writeln('<tr>');

   var r = i % 7;
   if(r == 0 || r == 6){
      document.writeln('<tr id="on">');
   }
   else{
      document.writeln('<tr id="off">');
   }
   document.writeln('<td>'+ str[i] +'</td>');
   document.writeln('</tr>');
}
document.writeln('</table>');
</script>
</body>
</html>
```

曜日名を配列として準備します

休日の処理です

平日の処理です

曜日名を表示します

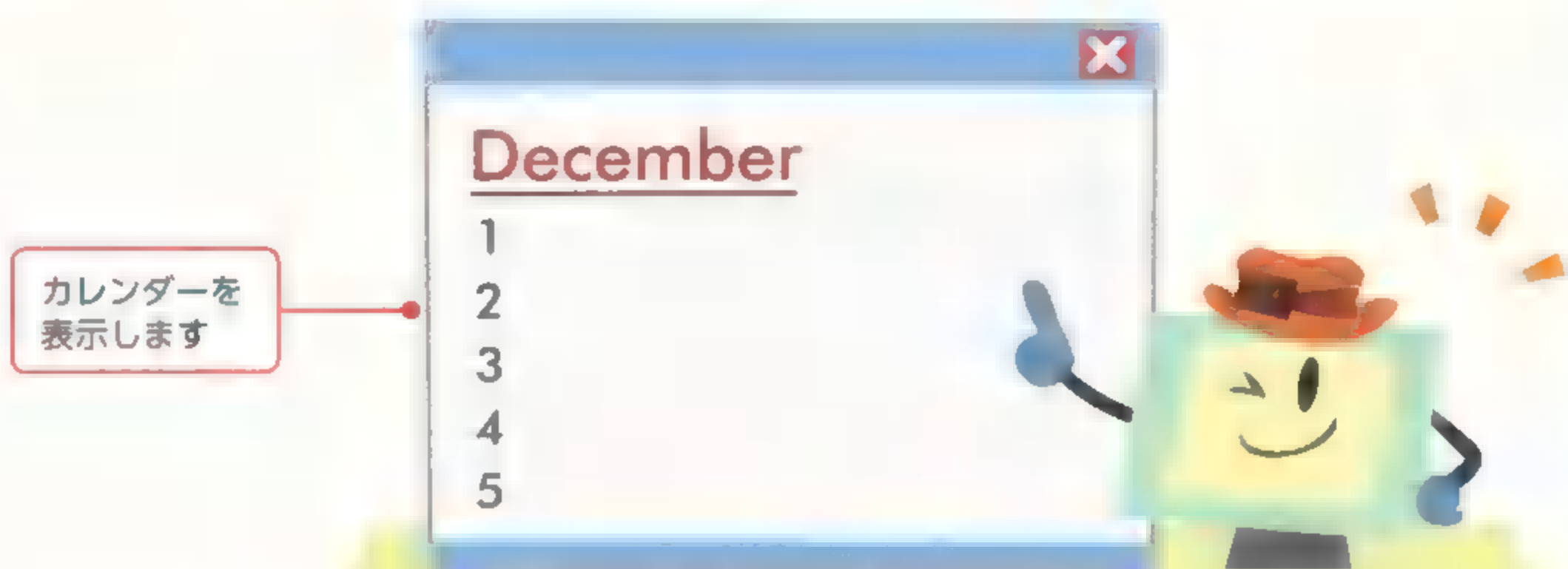
■Sample2-6-3の表示

データを用意する方法には、このほかにもさまざまな方法があります。大量のデータを用意するには、ファイルやデータベースを使って準備することもあります。

2-7 実践！ カレンダーを作ろう

チャレンジ！ オリジナルのカレンダーをつくろう

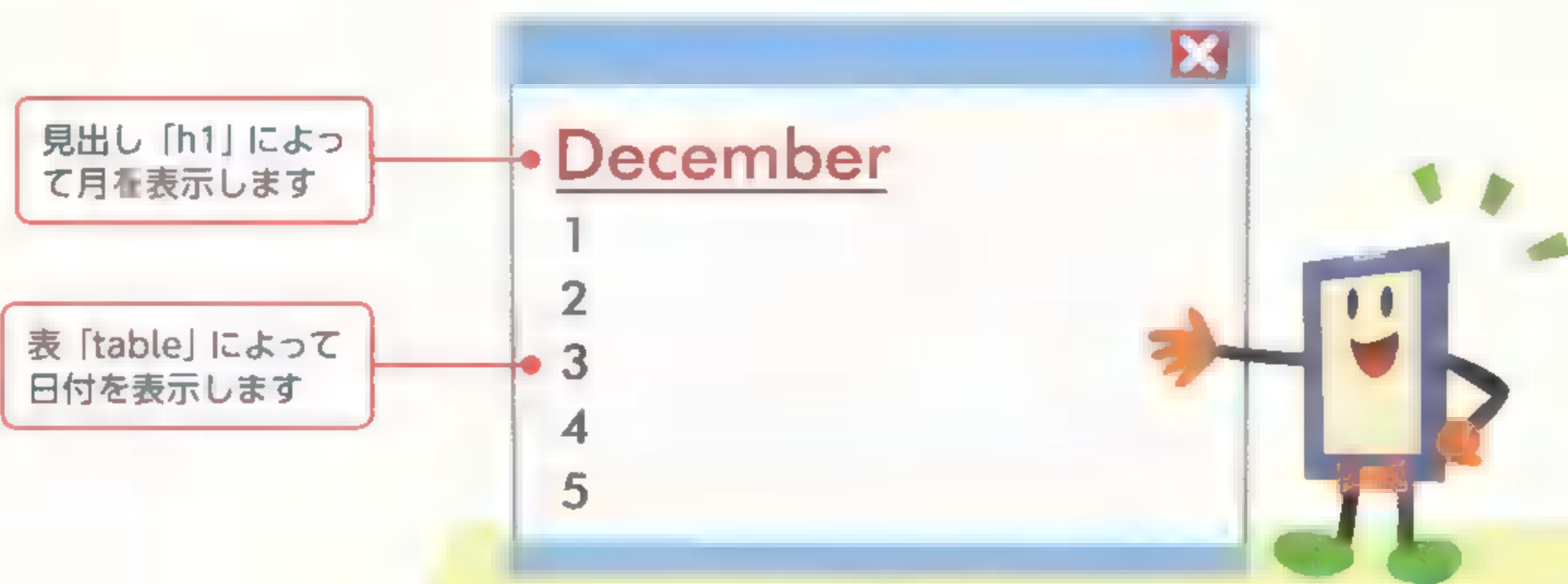
JavaScriptを使って、いろいろなWebページが作成ができるようになりました。それでは、これまでの知識を応用して、Webページに実践的な機能を付け加えていくことにしましょう。

この節では、Webページ上でカレンダーを作成してみることにします。今月分のカレンダーを作成し、Webページとして表示してみましょう。

■カレンダーを作成する

Step 1 概要を設計する

まず、カレンダーの概要について考えておきましょう。Webページを作成するにあたっては、どのような機能を作り上げていくのか、その概要を考えておくことが大切です。

ここでは、現在の月を大きく表示し、その下に日付の■列を表示することにします。月は「h1」要素を使って大きく表示し、日は「table」要素を使って表示することにします。

■カレンダーの表示の概要を考える

Step 2 コードを考える

日時を調べる

それでは、いよいよJavaScriptのコードを考えることにしましょう。日時はこれまでみてきたように、Dateオブジェクトから調べることができるでしょう。曜日名の配列データを用意することにします。

```
var str = ["Sun","Mon","Tue","Wed","Thu","Fri","Sat"];
```

曜日名を配列として準備します

月を調べる

次に、Dateオブジェクトから月を調べましょう。現在の月を見出しとして表示することにします。

```
document.writeln("<h1>");
var d = new Date();
var m = d.getMonth();
document.writeln(m+1);
document.writeln("</h1>");
```

月を調べます

日数分繰り返す

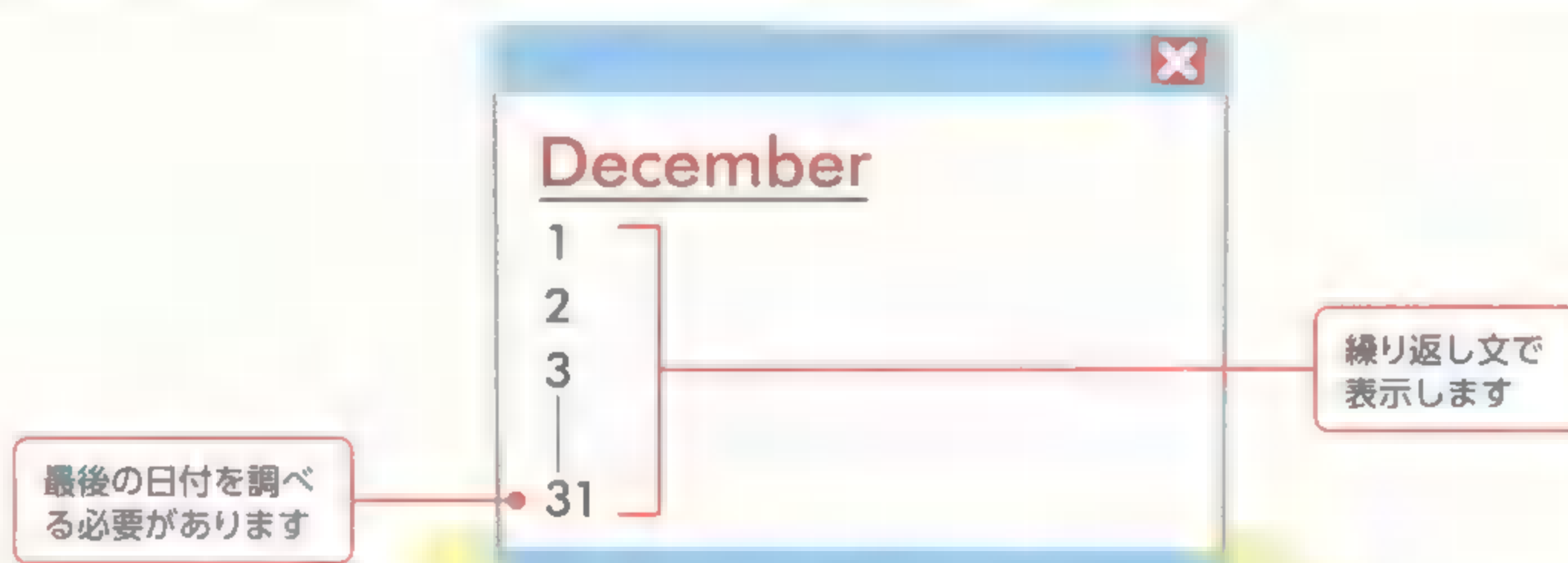
1か月に日の部分は30日前後ありますから、プログラムらしく繰り返し文を使って、かんたんに表示することにしましょう。

ここで注意しなければならないのは、月の最後の日のことでしょう。月の最後の日は月によって異なります。これは次の指定によって調べることができます。

```
var fd = new Date(d.getFullYear(), d.getMonth(), 1);
var f = fd.getDay();
var ld = new Date(d.getFullYear(), d.getMonth()+1, 0);

for(var i=1; i<=ld.getDate(); i++){
・・・
```

月の最後の日を調べます

月の最後の日を得ることができます

■日数分繰り返す

Step3 カレンダーを完成させる

それでは、ここで見てきたコードをもとにして、カレンダーを作成していきましょう。この章のこれまでの表示とは違い、カレンダーの表示用のスタイルシート（Cal.css）も作成してみます。

Sample2-7-1.html ■ カレンダーを作る

```
<!DOCTYPE html>
<html lang="ja">
<head>
<link rel="stylesheet" href="Cal.css">
<title>サンプル</title>
</head>
<body>
```

```
<script type="text/javascript">
var str = ["Sun","Mon","Tue","Wed","Thu","Fri","Sat"];
    document.writeln("<h1>");
    var d = new Date();
    var m = d.getMonth();
    document.writeln(m+1);
    document.writeln("</h1>");

    document.writeln("<table>");

    var fd = new Date(d.getFullYear(), d.getMonth(), 1);
    var f = fd.getDay();
    var ld = new Date(d.getFullYear(), d.getMonth()+1, 0);

    for(var i=1; i<=ld.getDate(); i++){
        document.writeln("<tr>");
        document.writeln("<td>");
        document.writeln(i);
        document.writeln("</td>");
        document.writeln("<td>");
        document.writeln(str[(f+i-1)%7]);
        document.writeln("</td>");
        document.writeln("</tr>");
    }
    document.writeln("</table>");

</script>
</body>
</html>
```

曜日名を配列として準備します

月を調べます

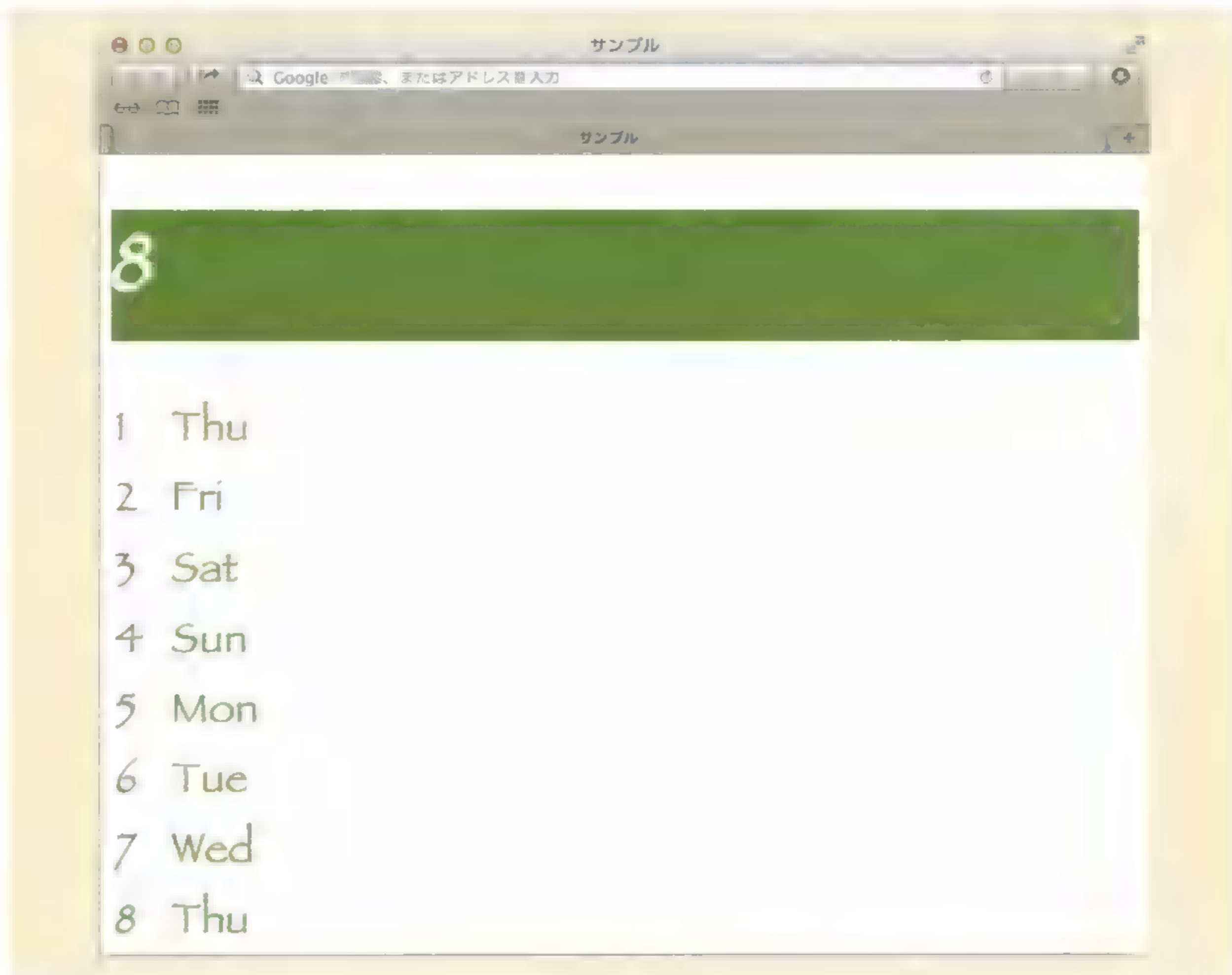
月の最後の日を調べます

月の最後の日を知ることができます

Cal.css ■ スタイルシート

```
body{
    background-color: #FFFFFF;
    color: #2e8b57;
    font-size: 2em;
    font-family: fantasy;
}
h1{
    background-color: #2e8b57;
    color: #FFFFF0;
    font-size: 2em;
    font-family: fantasy;
}
```

見出し「h1」によって月を表示します

■Sample2-7-1の表示

いかがでしょうか？　これまでの知識を活用して、実用的なWebページを作成することができたでしょうか？　学んだ知識を生かし、さまざまなデザインや構成をもつ、オリジナルのカレンダーを作成してみてください。

Tips

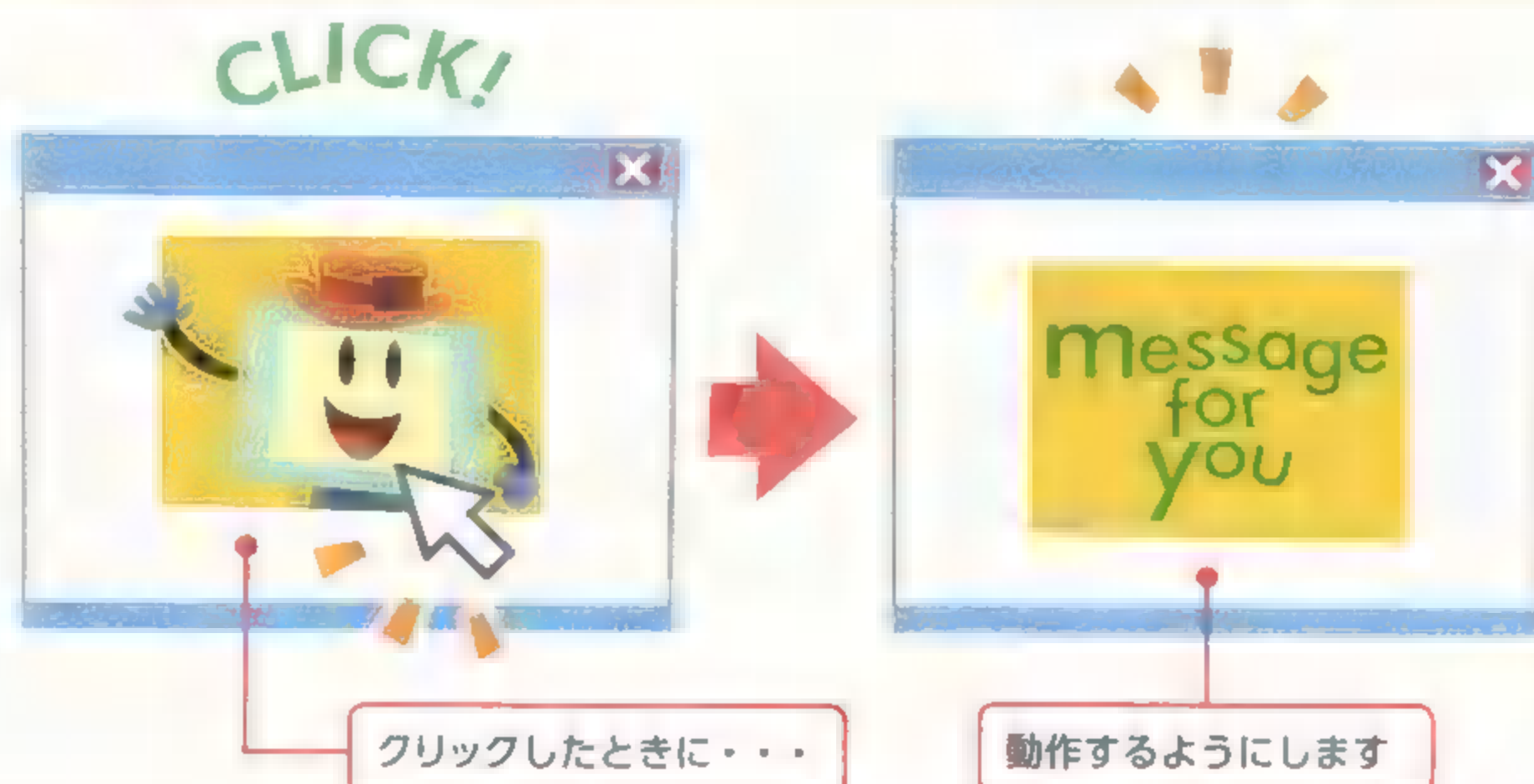
スタイルシートでレイアウトを行えば、同じ構成でも、いろいろなデザインのカレンダーを作成することができます。また、JavaScriptのコードを変更することによって、さまざまな要素を持つカレンダーを作成することができるでしょう。

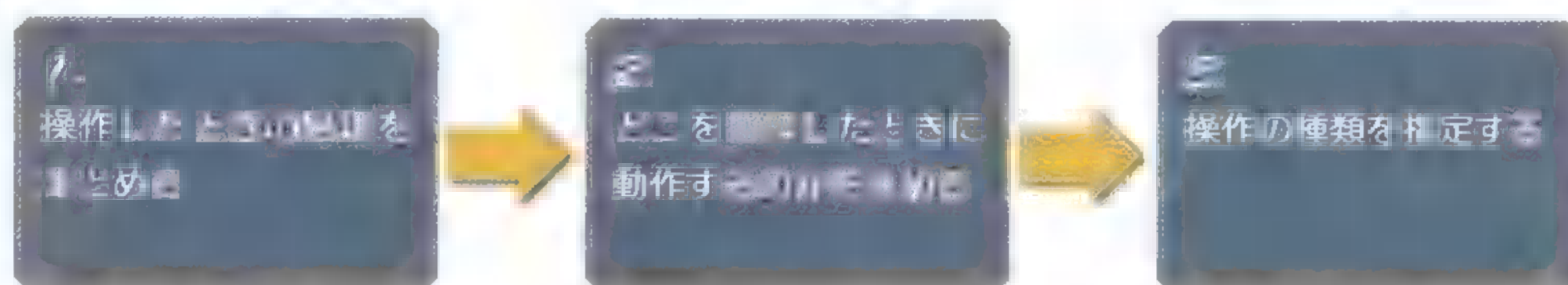
3-1 クリックで表示しよう

チャレンジ！ クリックで動作するようにしよう

JavaScriptを使ってWebページを動かしていくことができます。そこでこの節では、Webの画面を操作したときに動作するWebページを考えてみましょう。

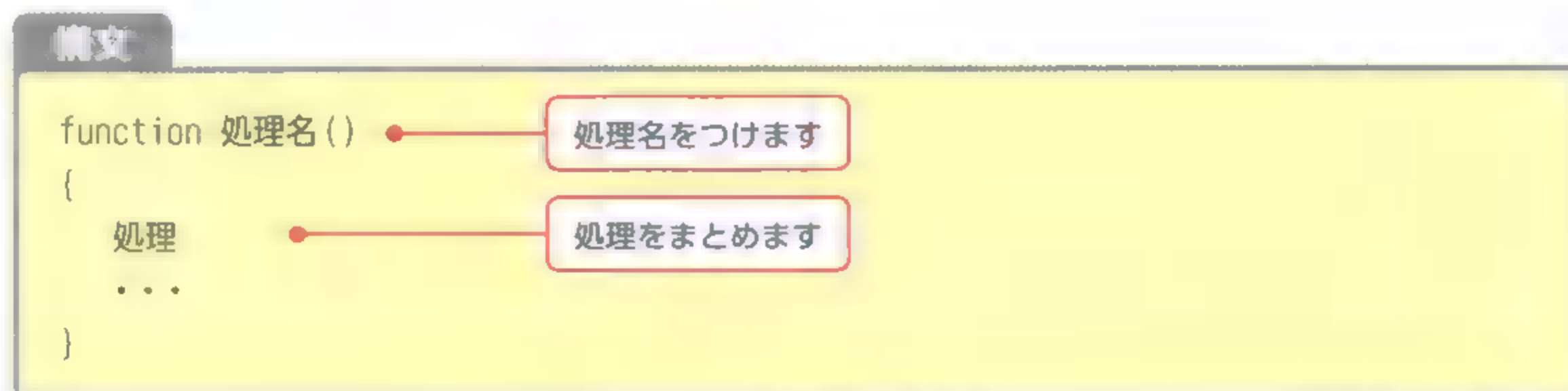
まず最初に、画像をクリックしたときにメッセージが表示されるプログラムを考えてみることにします。

■ クリックで動作するWebページ

ユーザーの操作で動作させるには

それでは、どうやってWebページを動かしたらいいでしょうか。ユーザーがWebページを操作したときにページを動かすためには、次のようにプログラムを作成します。

処理をまとめる

まず、ユーザーがページを操作したときに、どのように処理を行うのかをまとめておきます。このためには、functionという指定のあとに「処理名( )」と入力し、{ }内に順番に処理をまとめることになります。なお、「処理名」は変数の名前と同じように自分でつけてかまいません。

```
function 処理名()
{
    処理
    ・・・
}
```

処理名をつけます

処理をまとめます

たとえば、メッセージを表示する処理をまとめてみましょう。ここでは、welcome()という名前をつけて、ウィンドウにメッセージを表示する処理をまとめてみました。

```
function welcome()
{
    window.alert("ようこそ!");
}
```

メッセージを表示します

「処理名」には、変数と同じように自分で適当な名前をつけてかまいません。わかりやすい名前をつけるようにしましょう。

どこを操作したときに処理するのかを決める

次に、ユーザーがWebページ上のどの部分を操作したときに動作するのかを決めます。たとえば、画像を操作したときに動作するように考えてみましょう。「img」要素を操作したときに動作するように考えます。

```
<img src="Sample.jpg" ・・・
```

どこを操作したときに処理するのかを決めておきます

操作の種類を指定する

そして、どんな操作をしたときに処理をするのか、その操作のタイミングを指定します。たとえば、「クリックしたとき」に処理を行うためには、操作の対象をあらわす要素のタグ中に

onclick="処理名()"と指定します。onclickは「クリックしたとき」をあらわす指定です。

ここでは、画像をあらわすimgタグ中に、onclick="welcome()"と指定します。この指定によって、画像をクリックしたとき、先ほどwelcome()にまとめておいた処理が行われるようになるのです。

```
<img src="Sample.jpg" onclick= "welcome()"/>
```

操作のタイミングを指定します

クリック時の動作を確認しよう

それでは、クリックしたときにウィンドウメッセージを表示するページを作成してみましょう。なお、この章では次のスタイルシートを使うことにします。

Sample.css ■ 3章のスタイルシート

```
body{
   background-color: #000000;
   color: #FFFFF0;
   font-size: 2em;
   font-family: fantasy;
}
#smallimage{
   width: 40px;
   height: 30px;
}
```

Sample3-1-1.html ■ クリックで表示する

```
<!DOCTYPE html>
<html lang="ja">
<head>
<link rel="stylesheet" href="Sample.css">
<title>サンプル</title>
<script type="text/javascript">
function welcome()
{
   window.alert("ログインしてください。");
}
</script>
</head>
<body>
<img src="flower.jpg" onclick= "welcome()"/>
```

メッセージを表示する処理をまとめておきます

・・・

クリックしたときに処理します

```
</body>
</html>
```

■Sample3-1-1の表示

イベント時に動作させる

操作のタイミングには、いろいろな種類があります。たとえば、ページを読み込む際に処理が行われるようにするには、onload="処理名()"と指定します。ダブルクリックしたときに処理するには、ondblclick="処理名()"と指定します。これら操作のタイミングの種類は、イベントと呼ばれています。

次のようなイベントを指定することができますから、いろいろなタイミングをためしてみてください。

■イベントの[illegible]

イベント	説明
load	読み込み時
unload	ページの移動時
resize	サイズ変更時
click	クリック時
dblclick	ダブルクリック時
mousedown	マウスボタン下げ時
mouseup	マウスボタン上げ時
mouseover	マウスカーソルがあるとき
mouseout	マウスカーソルが離れたとき
mousemove	マウスカーソルが動いたとき

イベント	説明
keypress	キーを押したとき
keydown	キーを下げたとき
keyup	キーを上げたとき
focus	フォーカス取得時
blur	フォーカス喪失時
change	フォームの変更時
select	フォームの選択時
submit	フォームの送信時
reset	フォームのリセット時

Tips

マウスでクリックしたり、マウスカーソルをあてたりする操作は、Webページにとっては重要な出来事でしょう。プログラミングの世界では、このような出来事（event）を「イベント」と呼んでいるのです。

イベントの処理をしよう

それでは、実際にいろいろなイベントの処理を行い、動作を確認してみましょう。

ダブルクリック時に表示する

ondblclick="処理名()"と指定することで、ダブルクリック時に処理できるようになります。画像をダブルクリックしたときに、その画像を新しいウィンドウで表示するようにしてみましょう。

Sample3-1-2.html ■ ダブルクリックで表示する

```
<!DOCTYPE html>
<html lang="ja">
<head>
<link rel="stylesheet" href="Sample.css">
<title>サンプル</title>
<script type="text/javascript">
function welcome()
{
   window.open("animal.jpg", "サンプル", "width=100,height=100");
}
</script>
</head>
<body>
<img src="animal.jpg" ondblclick= "welcome()"/>
</body>
</html>
```

ダブルクリック時に処理します

Tips

Windowオブジェクトのopen()メソッドを使うと、新しいウィンドウを開くことができます。ここでは、開くファイル名、タイトル、ウィンドウの幅・高さを指定して、新しいウィンドウを表示しています。なお、Windowオブジェクトはウインドウ機能をまとめたもので、オブジェクトを作成しなくても「window」という変数で利用することができます。

■ Sample3-1-2の表示

ボタンのクリックで表示する

ボタンをあらわす「input」要素の中にonclick="処理名()"を記述することで、ボタンをクリックしたときに処理できるようになります。

Sample3-1-3.html ■ ボタンのクリックで表示する

```
<!DOCTYPE html>
<html lang="ja">
<head>
<link rel="stylesheet" href="Sample.css">
<title>サンプル</title>
<script type="text/javascript">
function welcome()
{
   window.open("green.jpg", "サンプル", "width=100, height=100");
}
</script>
</head>
<body>
<img src="green.jpg"/>
<input type="button" value="Push Button" onclick= "welcome()"/>
</body>
</html>
```

ボタンを・・・

クリックしたときに表示します

■Sample3-1-3の表示

関数

ここで処理をまとめるしくみは、**関数**（function）と呼ばれています。関数はイベントが発生したときに、{ }内にまとめた処理が順番に行われるしくみになっています。「関数」という言葉は、あまりなじみがない言葉かもしれませんが、プログラミングの世界ではよく使われる言葉です。処理や機能をまとめるためのしくみとして使われています。覚えておくと便利でしょう。

```
function welcome()
{
  window.alert("ログインしてください。");
}
```

関数と呼びます

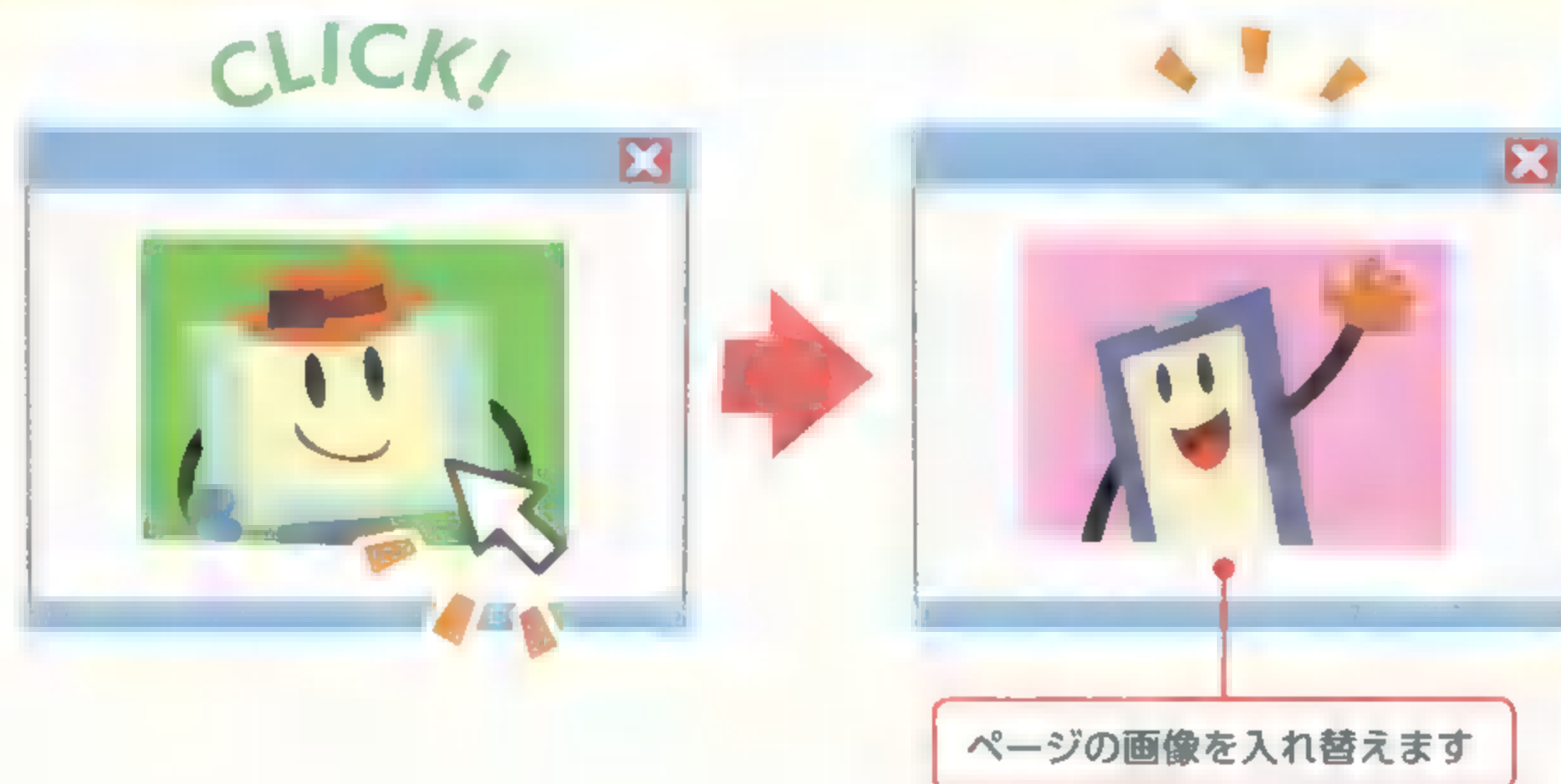
3-2 画像を入れ替えよう

チャレンジ！ 画像を入れ替えるには？

前の節では、クリックしたタイミングなどで、画像やメッセージを表示してしてみました。ただし、前の節のプログラムでは、Webページとは別のウィンドウを開いてメッセージや画像を表示しています。

クリックしたタイミングで、Webページ上のテキストや画像を差し替えるには、さらにいろいろな方法を学んで身につけていく必要があります。

そこで、今度はクリックしたタイミングで、Webページ上の画像を入れ替えてみることにしましょう。Webページ上の画像は、かんたんな方法で入れ替えることできます。

■ページ中の画像の入れ替え

画像を入れ替えるプログラム

クリックしたタイミングで、Webページ上の画像を入れ替えるには、次のページを作成します。

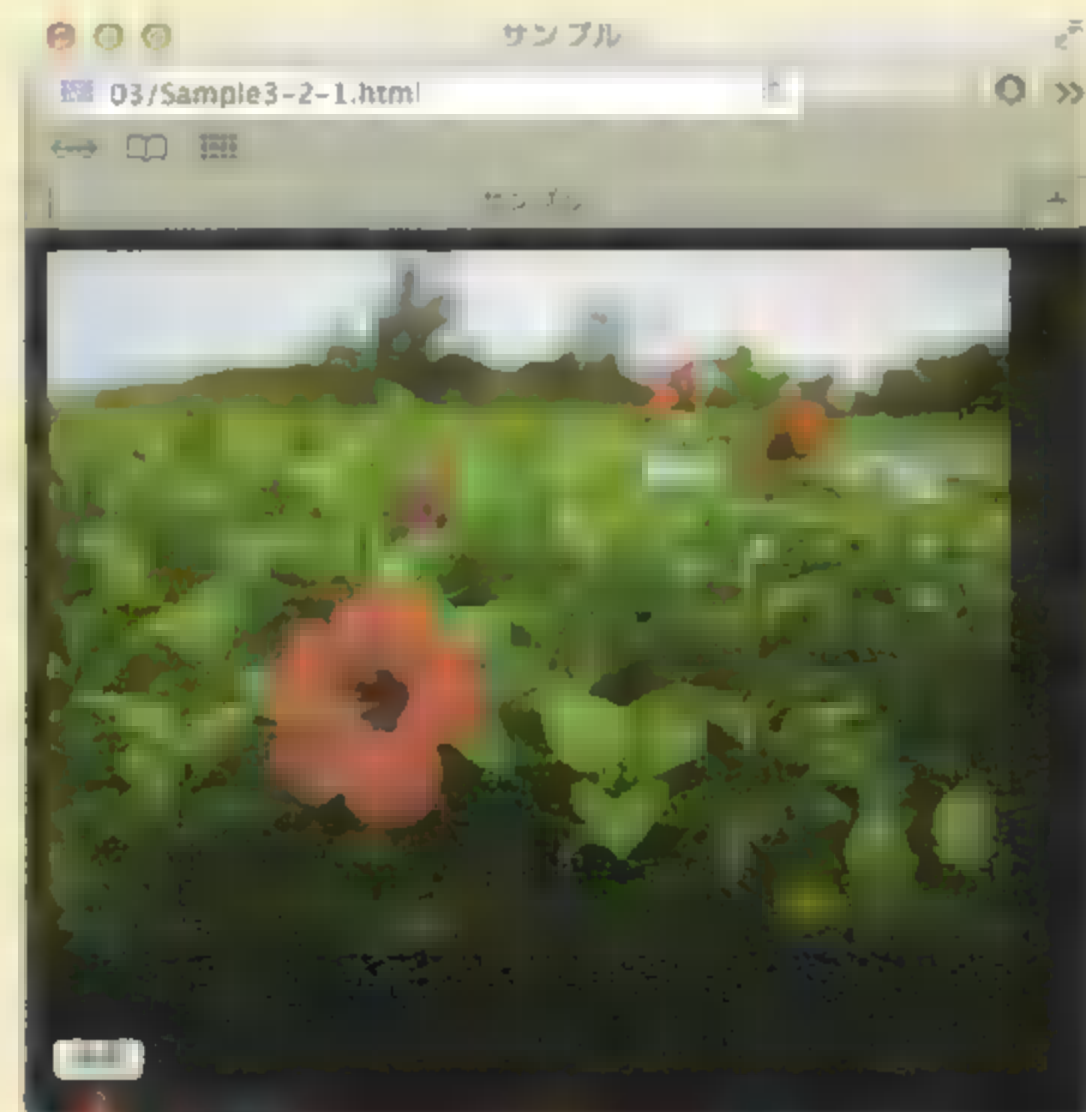
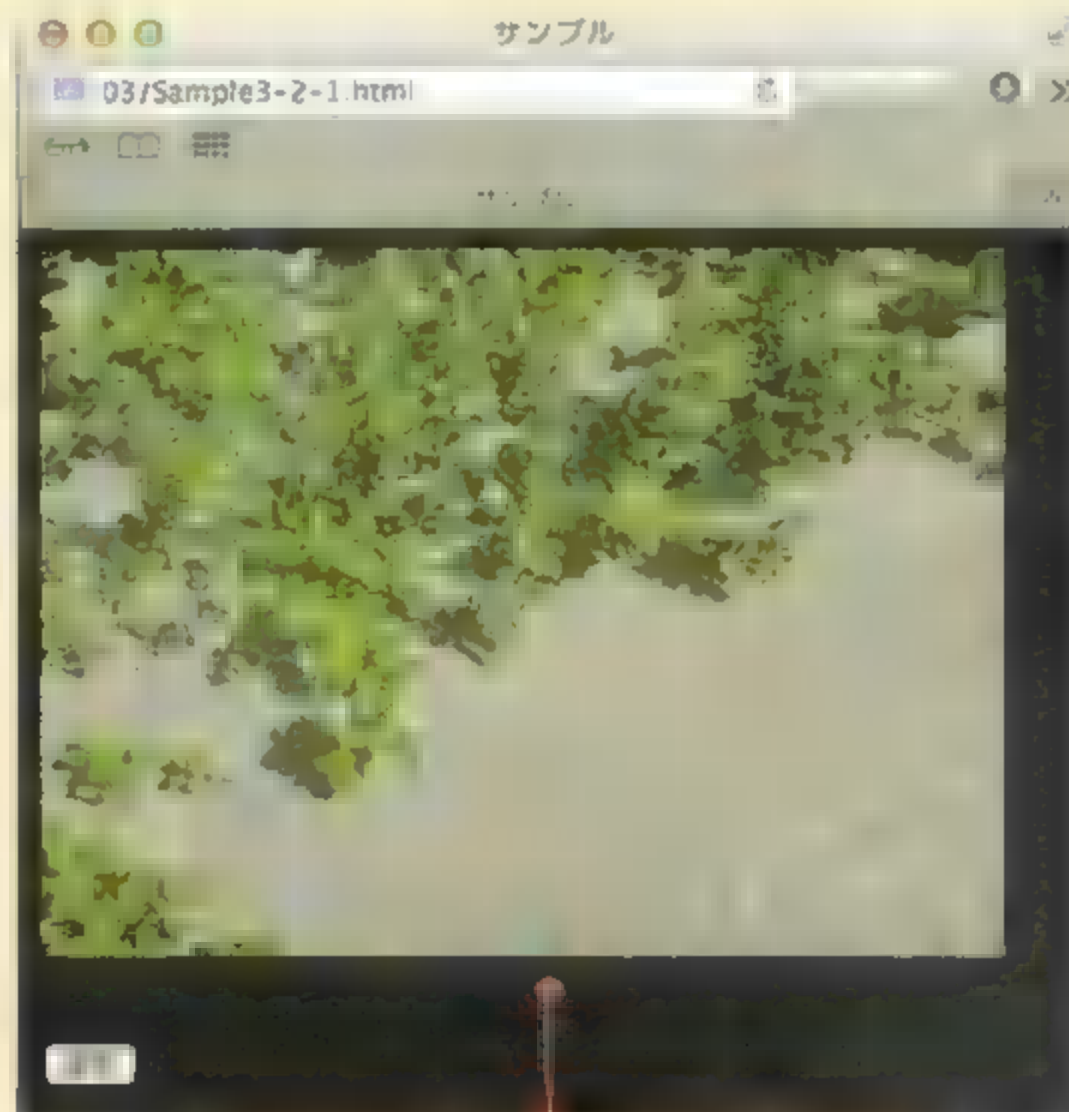
Sample3-2-1.html ■ 画像を入れ替える

```
<!DOCTYPE html>
<html lang="ja">
<head>
<link rel="stylesheet" href="Sample.css">
<title>サンプル</title>
<script type="text/javascript">
function change()
{
  document.image.src = "green.jpg";
}
</script>
</head>
<body>
<img name="image" src="flower.jpg">
<input type="button" value="変更" onclick="change()"/>
</body>
</html>
```

画像要素の名前を指定します

入れ替える画像を指定します

名前をつけておきます

■ Sample3-2-1 の表示

画像を入れ替えるためには?

画像を入れ替えるには、まず画像をあらわす「img」要素の「name」属性を使って、名前をつけておくことにします。ここでは、imageという名前をつけることにしました。

```
<img name="image" src="flower.jpg">
```

画像要素に名前をつけておきます

すると、document.名前.src="新しい画像ファイル名"と指定することで、画像だけを入れ替えることができるようになります。

つまり、ここでは次のように、flower.jpgからgreen.jpgに画像を入れ替えることができるわけです。

```
document.image.src = "green.jpg";
```

新しい画像に入れ替えます

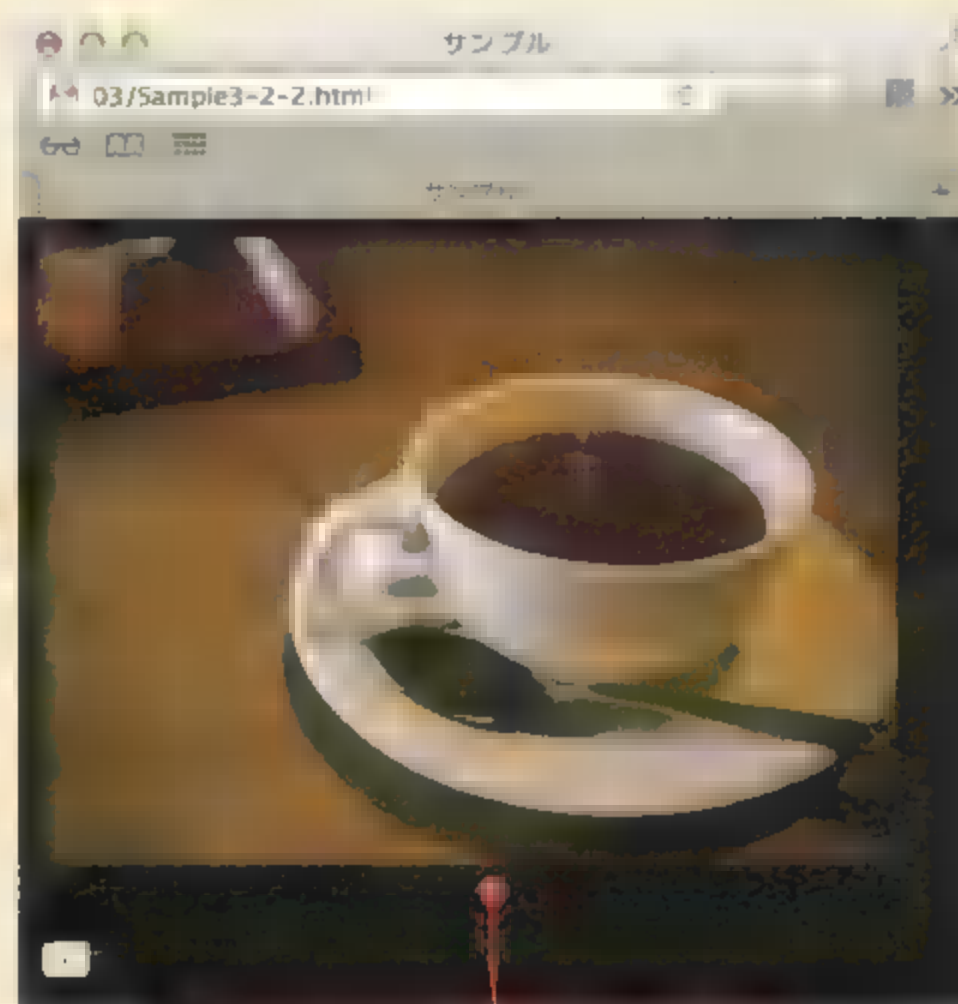
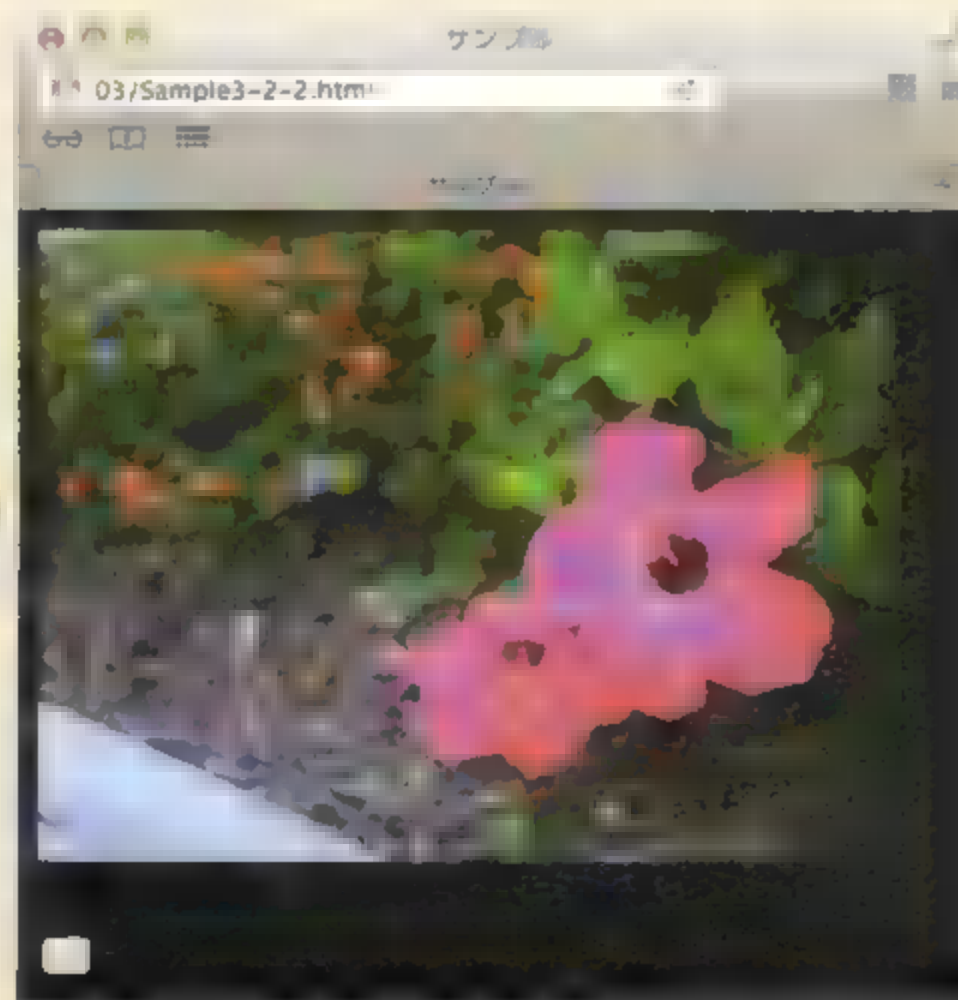
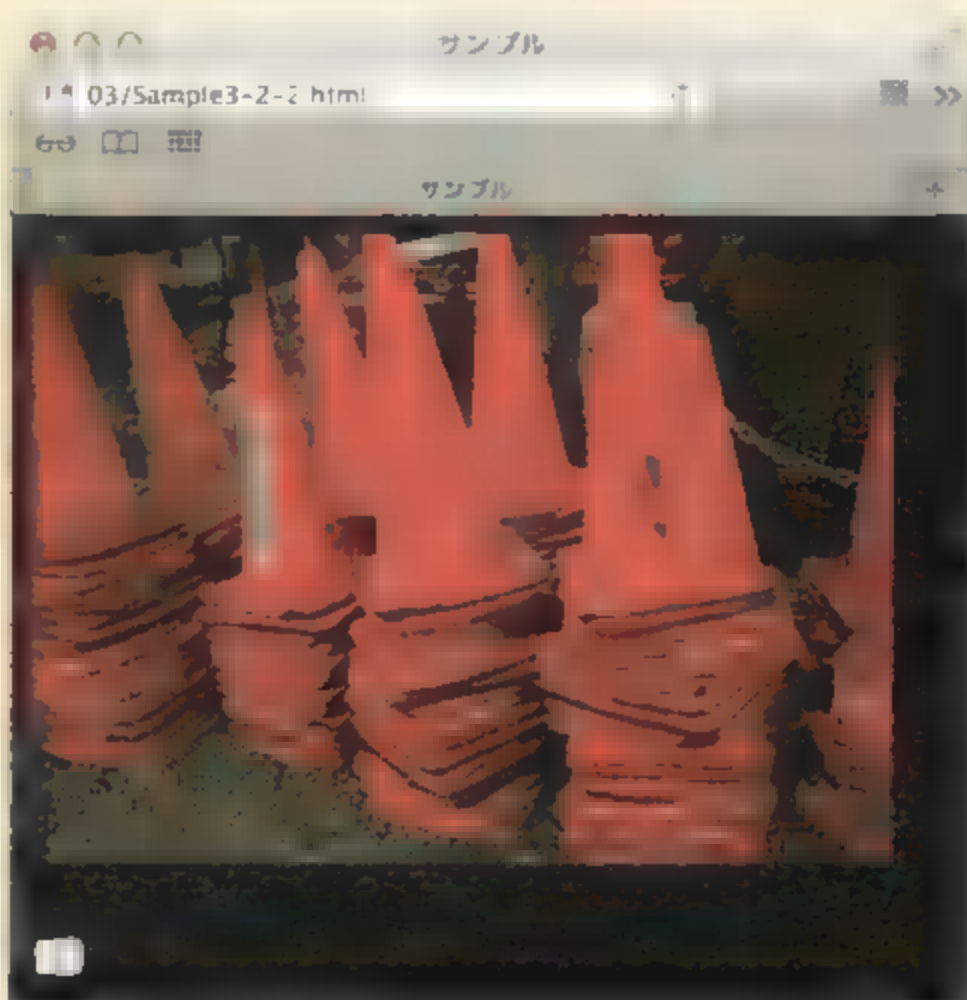
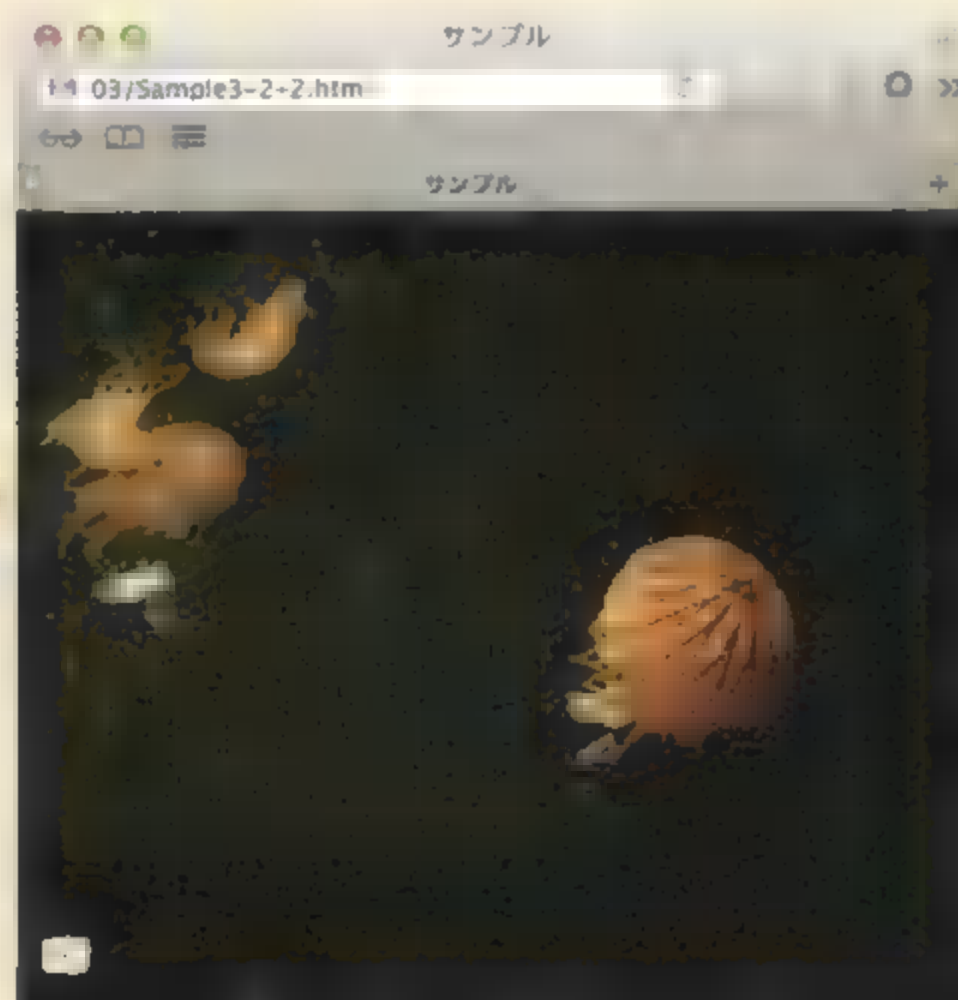
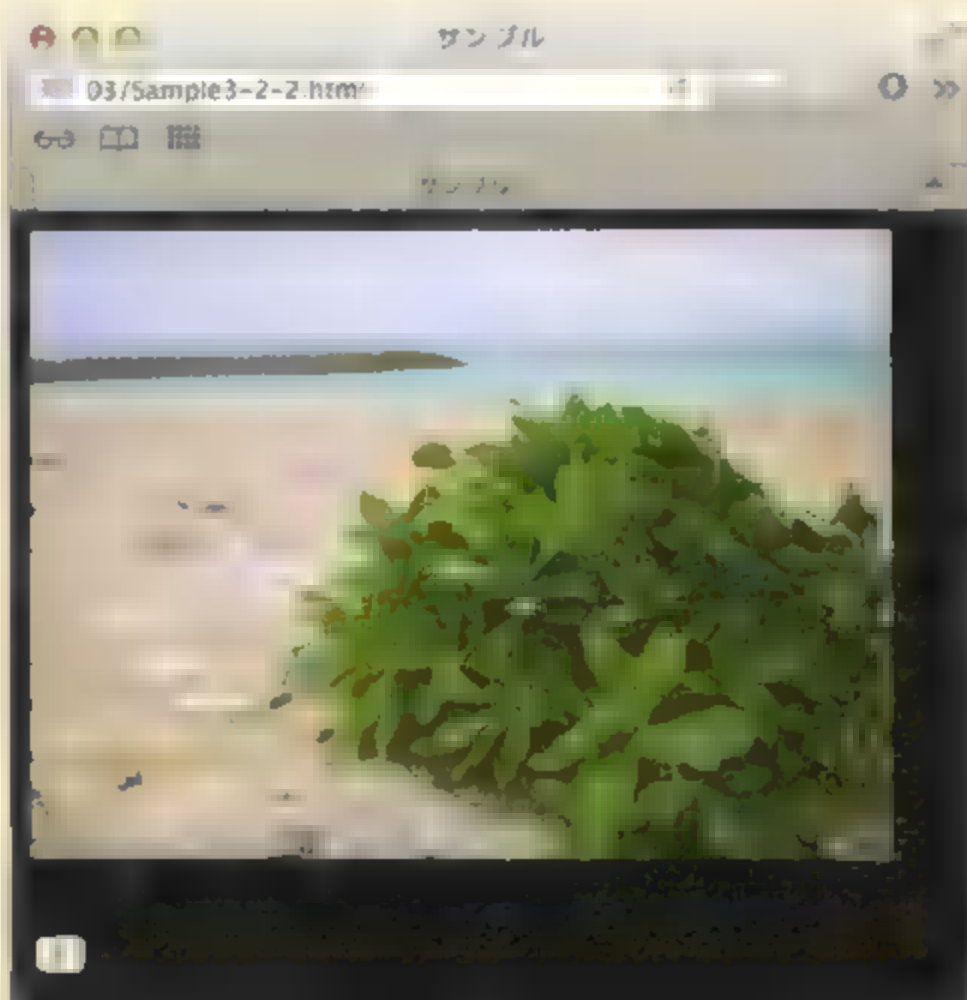
応用! 画像を順に表示しよう

それでは、写真を7枚用意しておき、クリックしたときに順番に画像を表示していくことにしましょう。

Sample3-2-2.html ■ 画像を順番に表示する

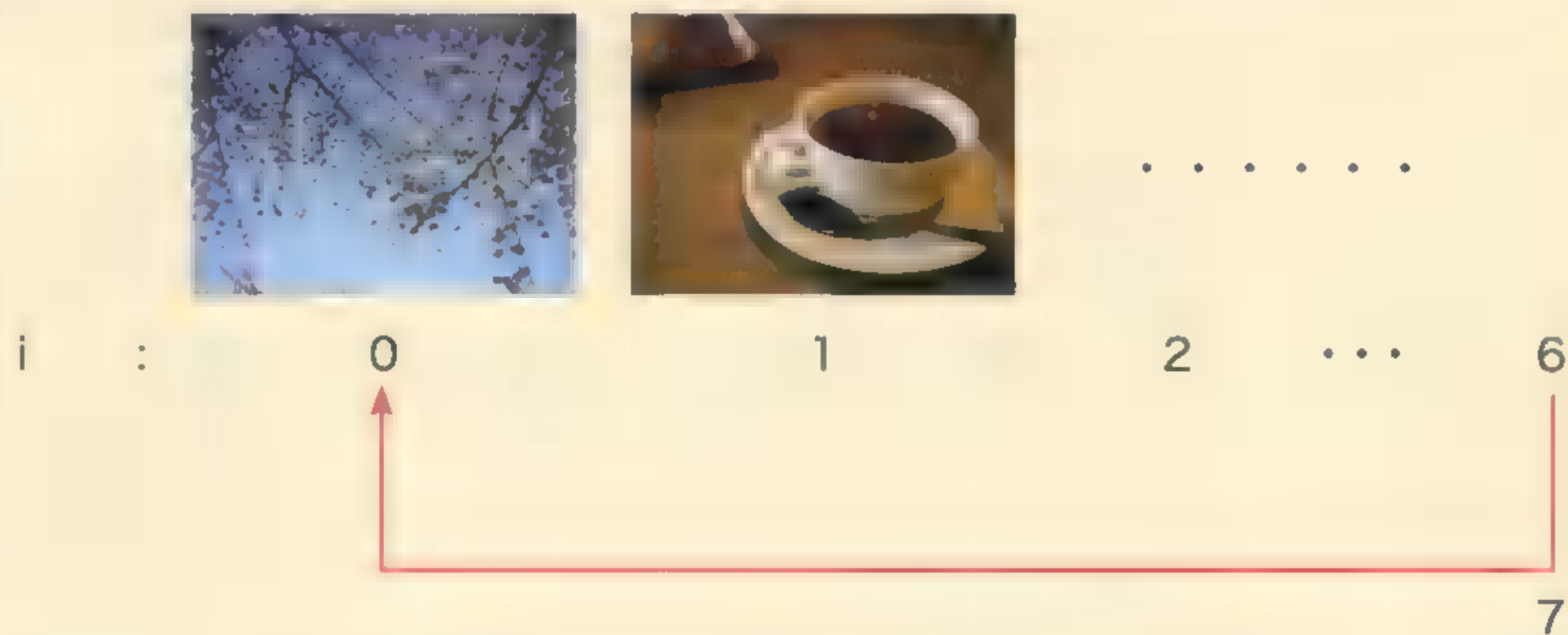
```
<!DOCTYPE html>
<html lang="ja">
<head>
<link rel="stylesheet" href="Sample.css">
<title>サンプル</title>
<script type="text/javascript">
var i = 0;
function change()
{
   i++;
   if(i == 7) i=0;
   document.image.src = "pic" + i +".jpg";
}
</script>
</head>
<body>
<img name="image" src="pic0.jpg">
<form>
<input type="button" value="●" onclick="change()"/>
</form>
</body>
</html>
```

- var i = 0; — 処理がはじまる前に番号を0としています
- i++; — 次の画像とします
- if(i == 7) i=0; — 存在しない番号になったら番号を0に戻します
- document.image.src = ... — 画像を表示します

■ Sample3-2-2の表示

ここでは、順番にpic0、pic1・・・となるように変数iをふやしています。iが7であれば、0に戻して最初の画像を表示するようにしています。

```
var i = 0;
function change()
{
   i++;
   if(i == 7) i=0;
   document.image.src = "pic" + i +".jpg";
}
```

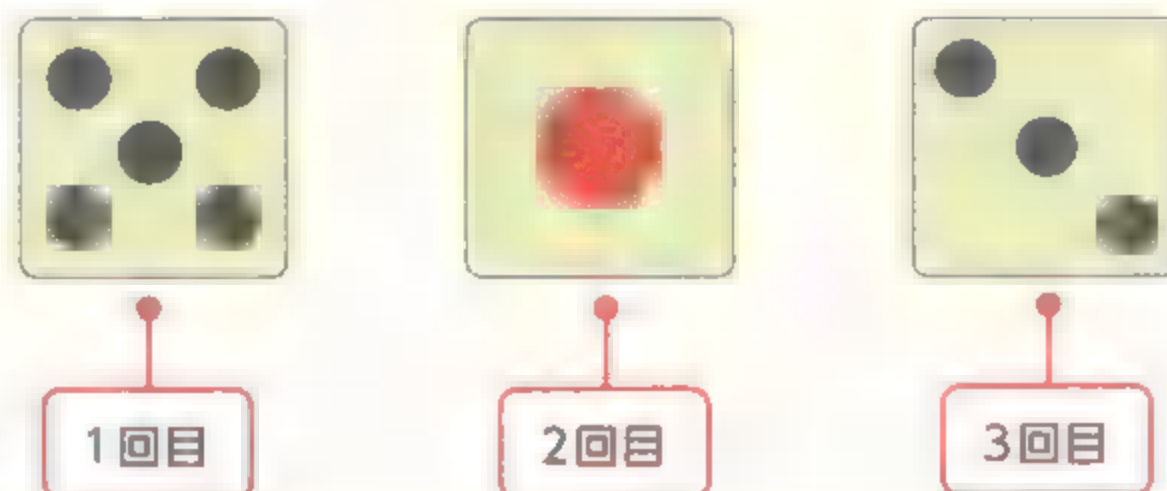
- `var i = 0;` ― 処理が始まる前に番号を0としています
- `i++;` ― 次の画像とします
- `if(i == 7) i=0;` ― 存在しない番号になったら番号を0に戻します
- `document.image.src` ― 画像を表示します

■画像を次々と表示する

なお、この変数iは、処理がはじまる前に、処理の外側で準備しています。処理がいったん終了しても、変数の値が引き続き記憶され続けるようにするためには、処理の外側で変数を準備する必要があります。

3-3 ランダムに表示しよう

チャレンジ!! サイコロの目を表示しよう

　サイコロをふると、1回目は5の目が、2回目は1の目というように、ランダムに目を出すことができます。こんなふうに、Webページでランダムに数字や画像を表示したい場合があります。そこでこの節では、ランダムに画像を表示する方法に挑戦してみましょう。

■ランダムな表示

ランダムに画像を表示する

　ランダムに画像を表示するには、次のようにプログラムを作成します。ボタンを押すと、1～6の目のいずれかがランダムに表示されます。

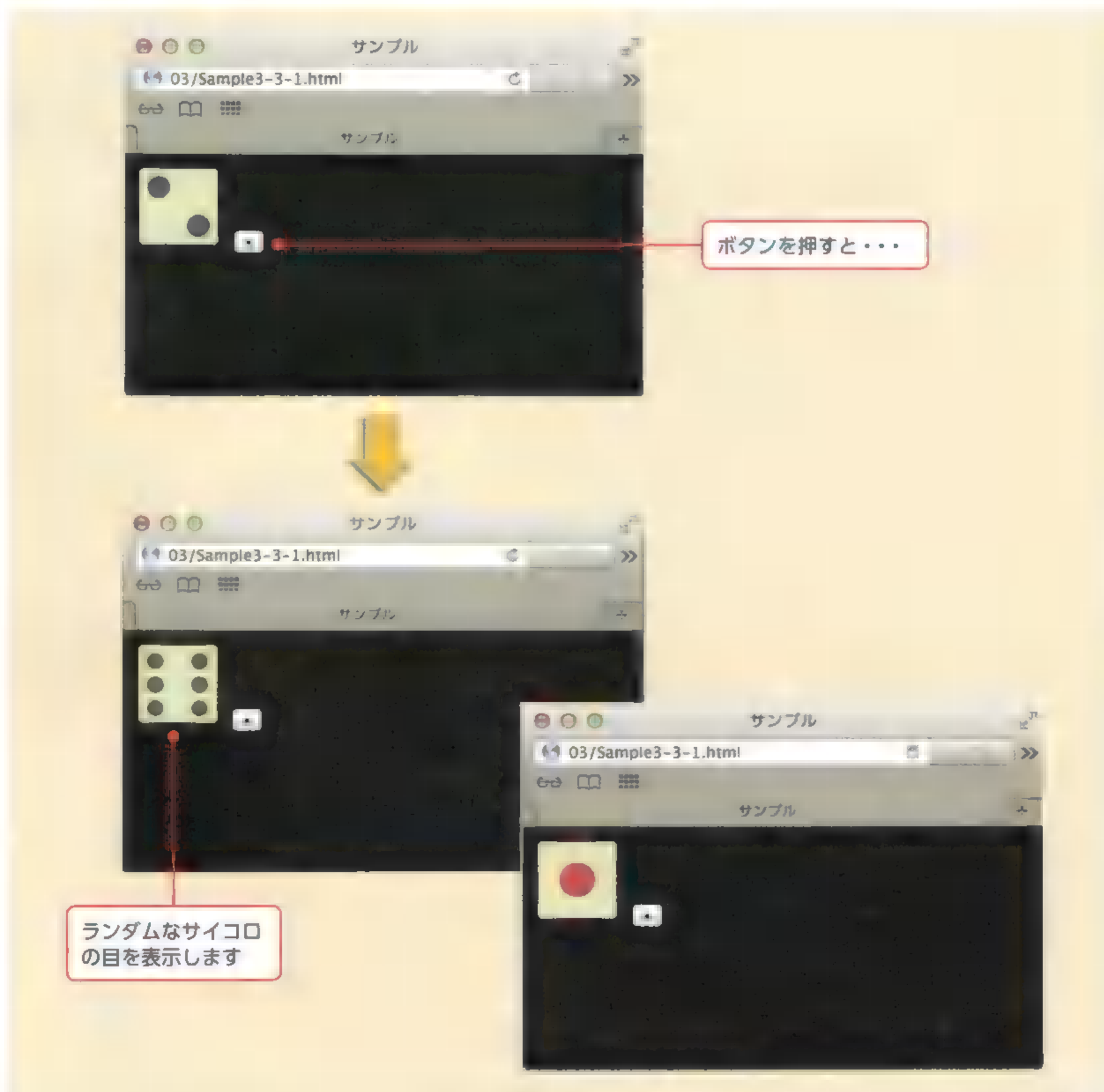
Sample3-3-1.html ■ ランダムに表示する

```
<!DOCTYPE html>
<html lang="ja">
<head>
<link rel="stylesheet" href="Sample.css">
<title>サンプル</title>
<script type="text/javascript">
function welcome()
{
   var r = parseInt(Math.random() * 6) + 1;
   document.dice.src = "dice" + r + ".jpg";
}
```

1～6のランダムな[illegible]を作成します（var r = parseInt(Math.random() * 6) + 1;）

ランダムなサイコロの目を表示します（document.dice.src = "dice" + r + ".jpg";）

```
</script>
</head>
<body>
<img src="dice1.jpg" name="dice">
<input type="button" value="●" onclick= "welcome()"/>
</body>
</html>
</body>
</html>
```

■Sample3-3-1 の表示

ランダムに表示するには？

■～●までの整数（●を含む）をランダムに表示するには、次のようにMathオブジェクトのrandom()メソッドを使います。

```
parseInt(Math.random() * ● ) + ■;
```

■～●のランダムな[illegible]を作成し[illegible]す

たとえば、ここでは1～6までをランダムに得るために、次のように指定しているわけです。

```
var r = parseInt(Math.random() * 6) + 1;
```

1～6のランダムな[illegible]を作成し[illegible]す

[illegible]

Math.random()は、Mathオブジェクトのメンバで、乱数と呼ばれるランダムな小数（0以上1未満）を求めることができます。
また、parseInt()は、小数を整数に変換する機能をもちます。random()で求めた小数を整数に変換しています。

Mathオブジェクト

Column

Mathオブジェクトでは、random()メソッドのほかにも次のようなメンバを利用することができます。Mathオブジェクトは、数学の計算をする機能をまとめたオブジェクトとなっています。

■Mathオブジェクトの主なメンバ

[illegible]	[illegible]
ceil(num)	切り上げて整数を取得
floor(num)	切り捨てて整数を取得
round(num)	四捨五入で整数を取得
abs(num)	絶対値を取得
max(num1, num2)	最大を取得
min(num1, num2)	最小を取得
pow(num, p)	p乗を取得
sqrt(num)	平方根を取得
sin(num)	サインを取得
cos(num)	コサインを取得
tan(num)	タンジェントを取得
random()	0以上1未満の乱数を取得
PI	π

応用1 ランダムな数を使いこなそう

ランダムな数値を得ることができればとても便利です。工夫次第でさまざまなWebページを考えることができるでしょう。

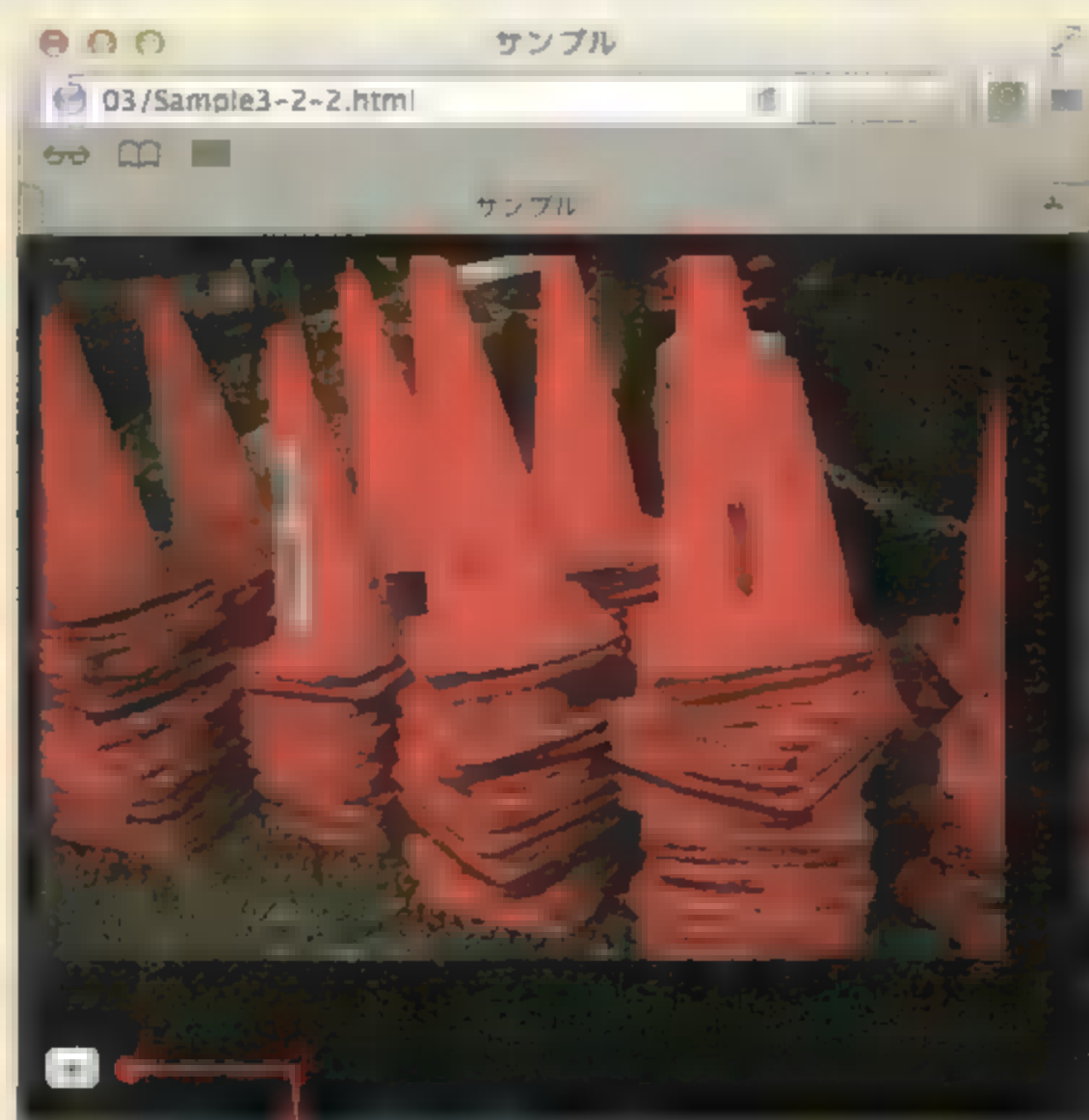
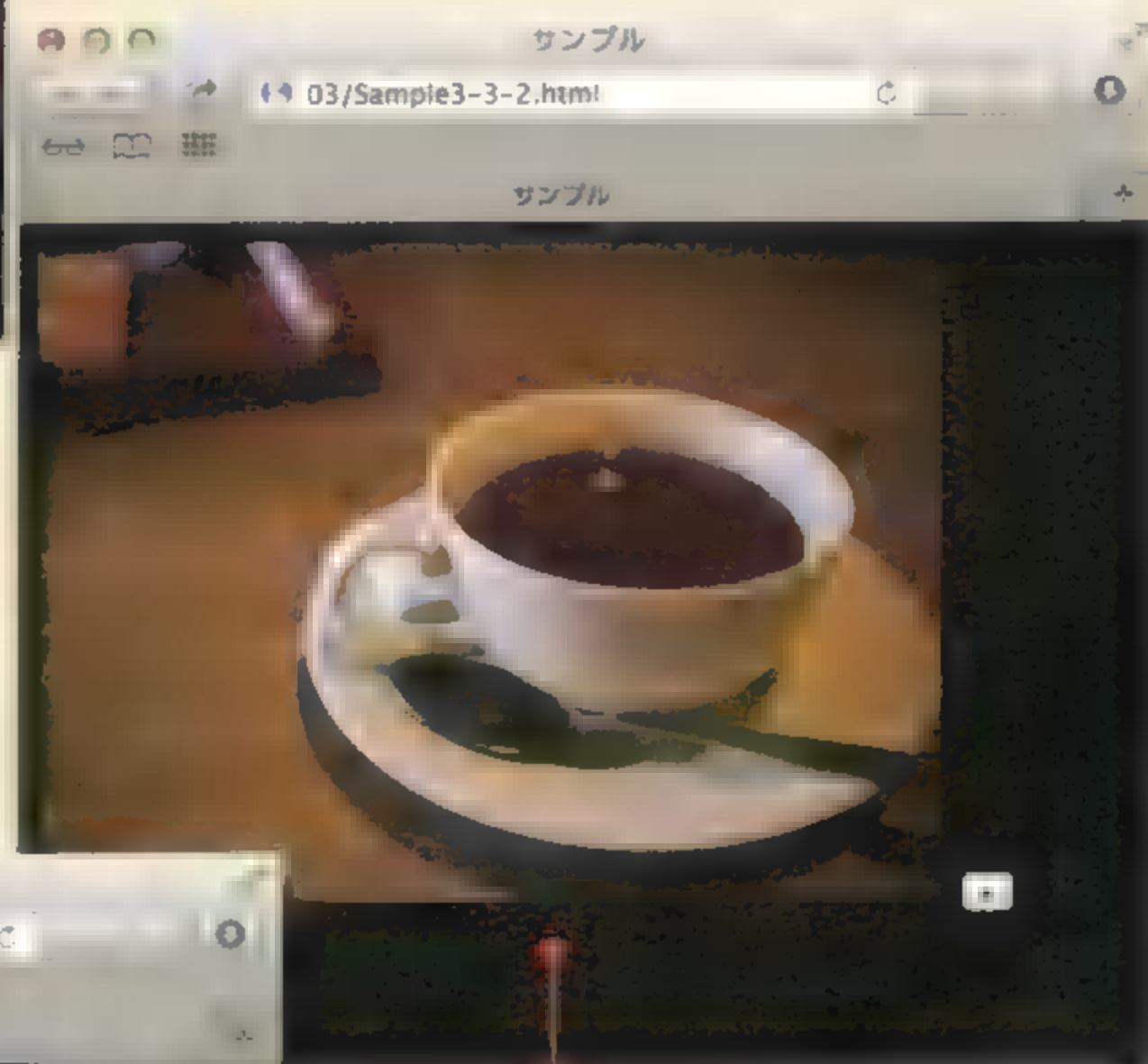
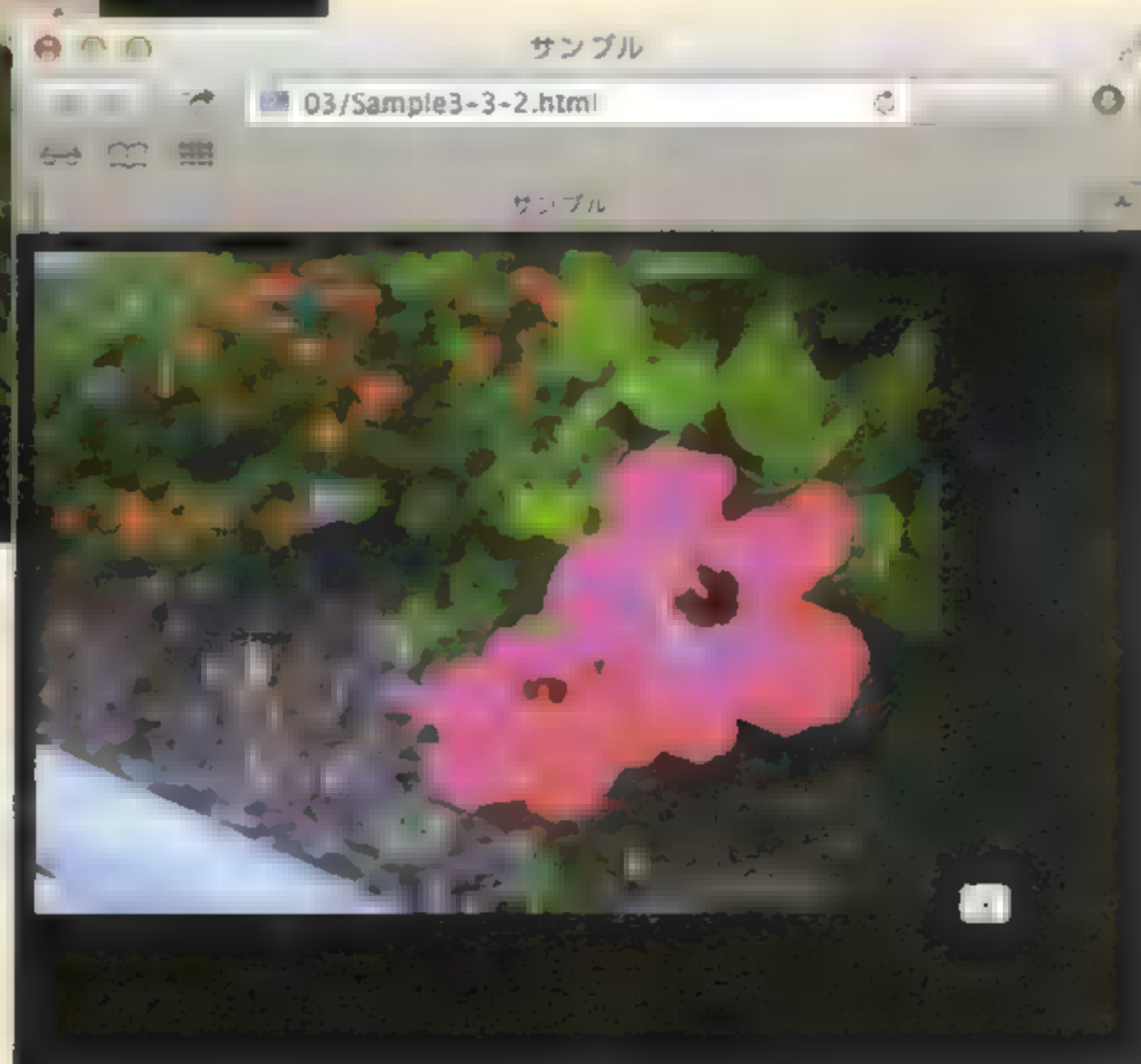
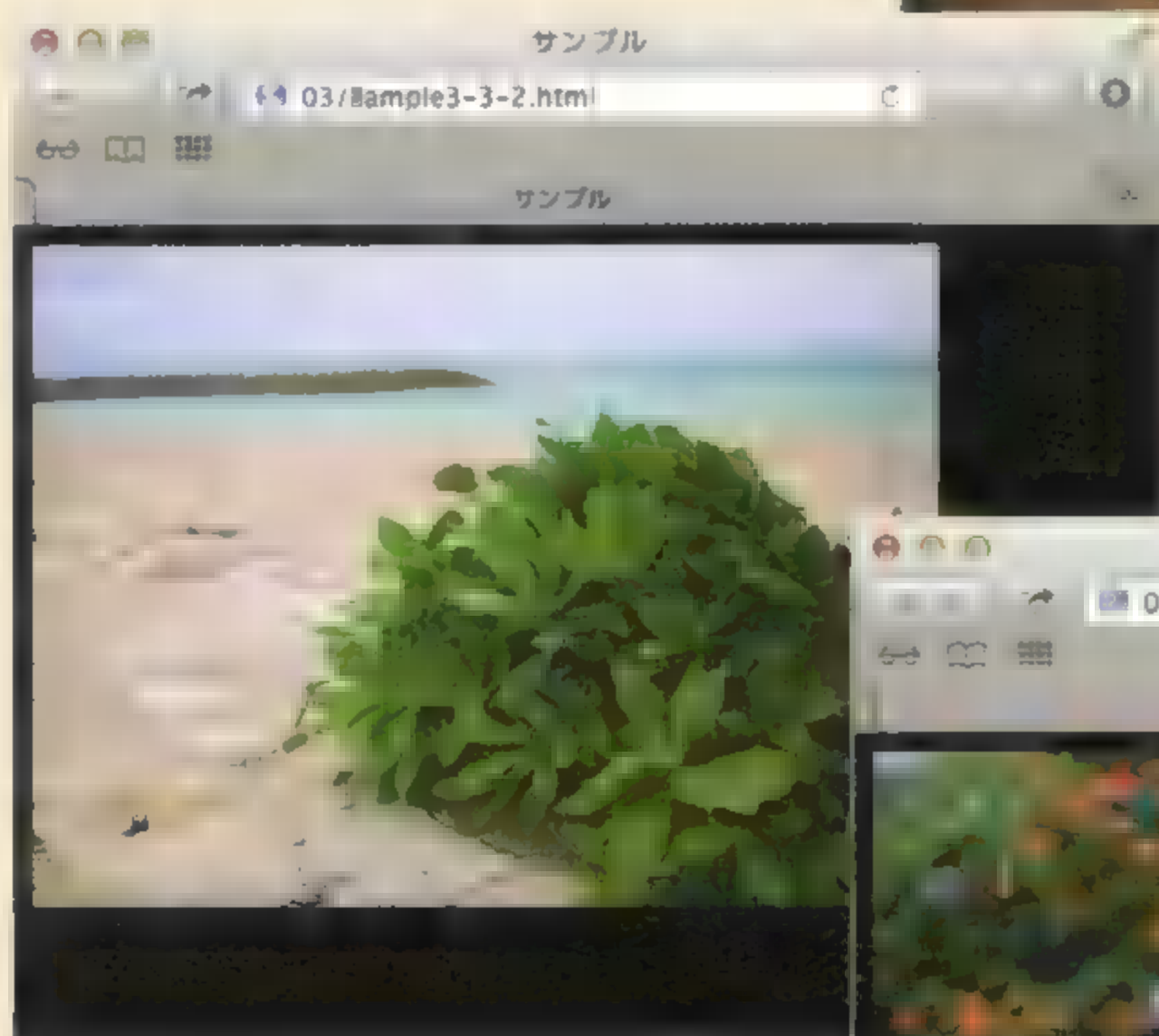
ランダムに入れ替える

画像をクリックするたびにランダムに表示してみましょう。今度はサイコロの目の画像のかわりに写真画像を使います。同じプログラムでも、画像を差し替えることで、いろいろなWebページに応用することができるのです。

Sample3-3-2.html ■ 写真をランダムに表示する

```
<!DOCTYPE html>
<html lang="ja">
<head>
<link rel="stylesheet" href="Sample.css">
<title>サンプル</title>
<script type="text/javascript">
function welcome()
{
   var r = parseInt(Math.random() * 6) + 1;
   document.pic.src = "pic" + r + ".jpg";
}
</script>
</head>
<body>
<img src="pic0.jpg" name="pic">
<input type="button" value="●" onclick= "welcome()"/>
</body>
</html>
```

pic1～pic6をランダムに表示します

■Sample3-3-2 の表示

見出しテキストをランダムに表示する

次に、ページを表示するたびに、3種類の見出しテキストがランダムに表示されるようにしてみましょう。タイトルメッセージを配列にしておき、この添字をランダムに指定することにします。

なお、このプログラムでは、タイトル画像となるtitle.jpgを用意し、次のスタイルシートTitle.cssを使います。

Title.css ■ 見出し用スタイルシート

```
body{
    background-color: #000000;
    color: #FFFFFF;
    font-size: 2em;
    font-family: fantasy;
    background-image: url(title.jpg);
    background-repeat: repeat-x;
}
h1{
    color: #FFFFFF;
    font-size: 4em;
    font-family: fantasy;
}
```

Sample3-3-3.html ■ 見出しをランダムに表示する

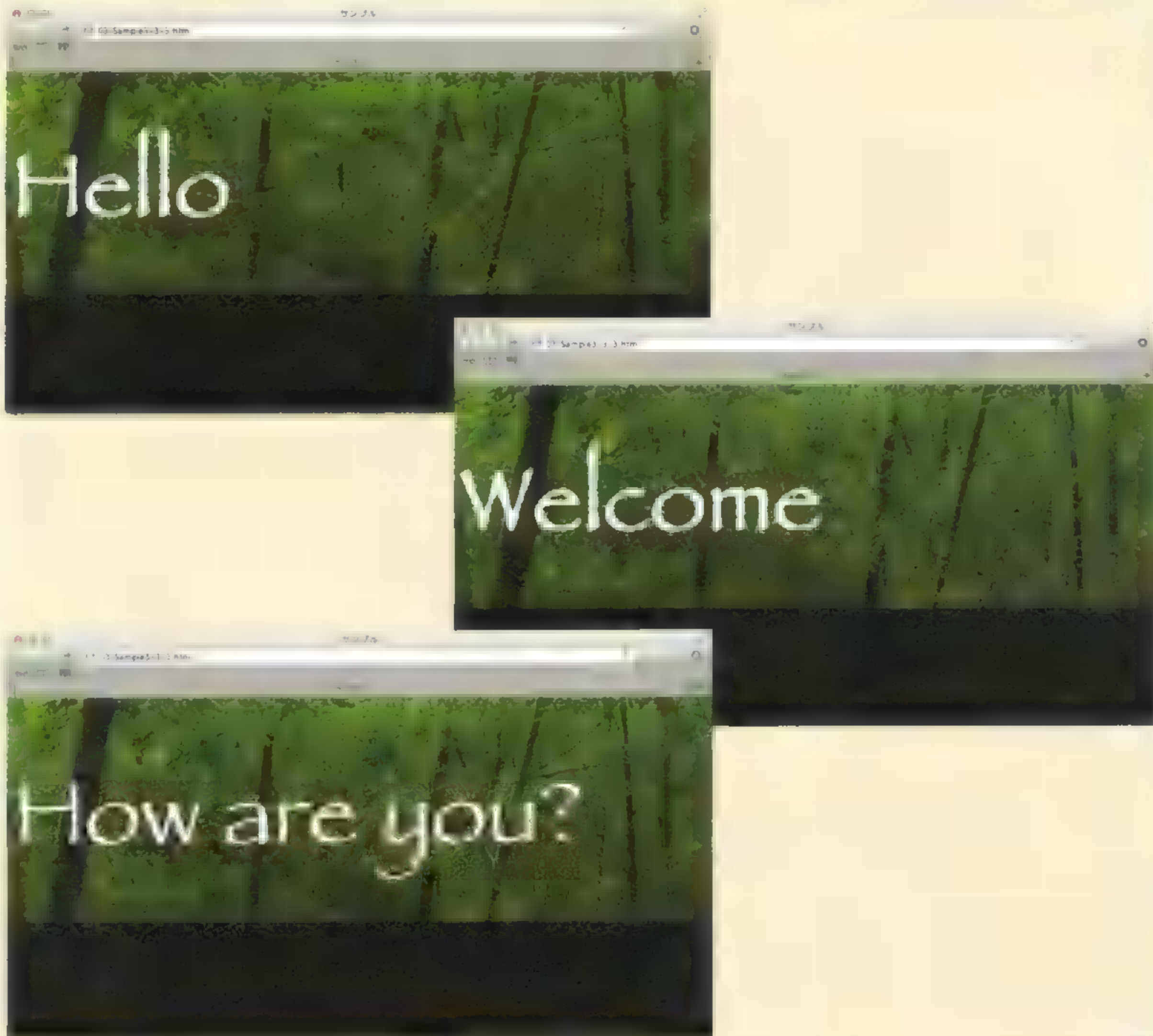
```
<!DOCTYPE html>
<html lang="ja">
<head>
<link rel="stylesheet" href="Title.css">
<title>サンプル</title>
</head>
<body>
<h1>
<script type="text/javascript">
var str = ["Welcome", "Hello", "How are you?"];
var r = parseInt(Math.random() * 3);
document.writeln(str[r]);
</script>
</h1>
</body>
</html>
```

見出しテキストを[illegible]で準備します

0以上3未満のランダムな[illegible]を作成します

ランダムな見出しテキストを作成します

3 JavaScriptで動かそう！

■Sample3-3-3の表示

このプログラムでは、起動するたびに違うタイトル画像が表示されます。Webサイトを開発するときにも役に立つプログラムとなるでしょう。

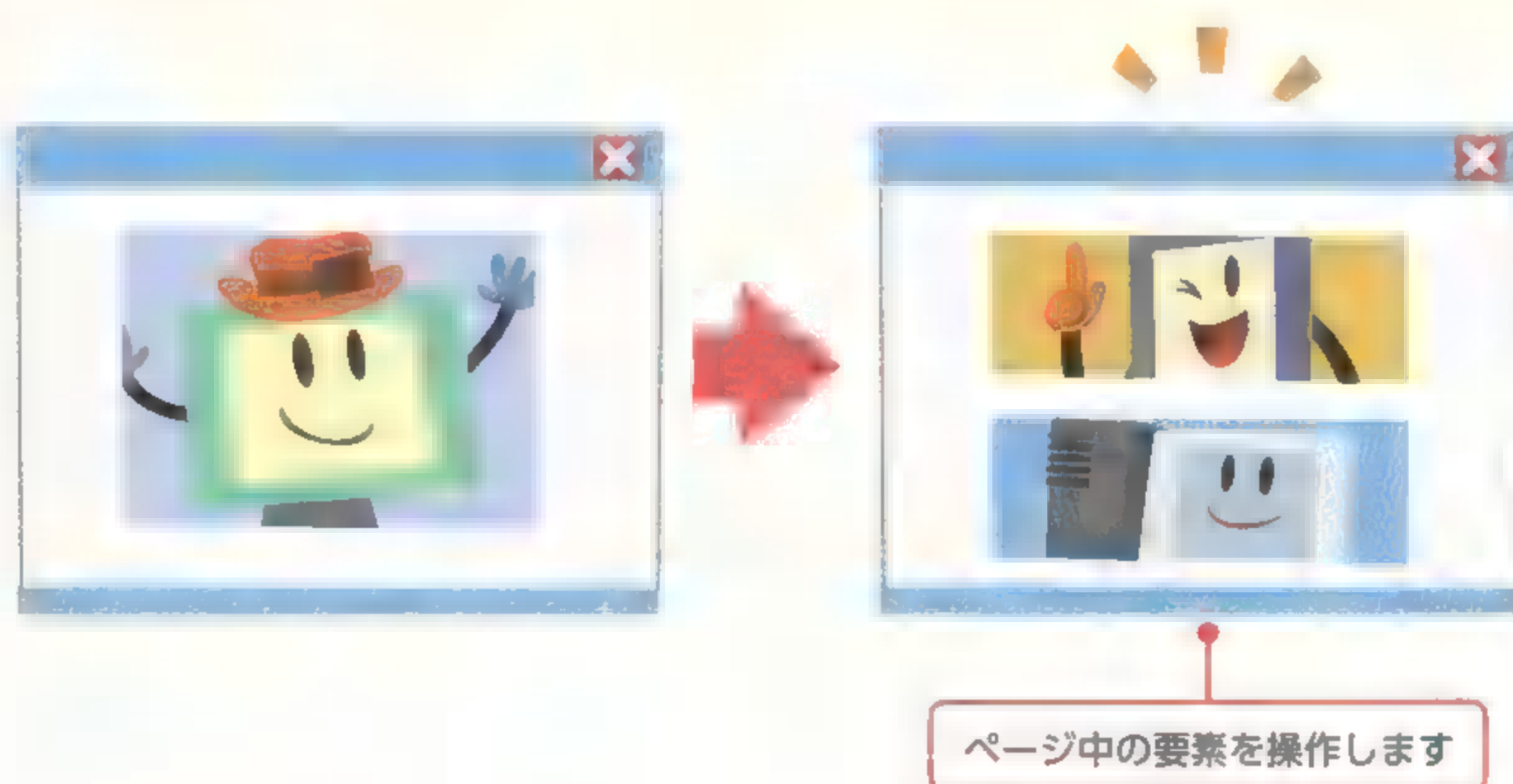
Tips

このページでは、タイトルテキストを入れ替えているのではなく、Documentオブジェクトのwriteln()メソッドを使って、最初にページ全体を作成しています。テキストだけを差し替える方法は、次の節で紹介しましょう。

3-4 ページを操作しよう

チャレンジ！ ページの一部を入れ替える

Webページの要素をよりきめこまかく操作するには、HTML文書を操作するための高度な知識が必要になります。そこで、今度はWebページを操作しながら、要素をきめ細かく操作する方法を学ぶことにしましょう。

■ ページ中の要素を操作する

ページの要素を操作する

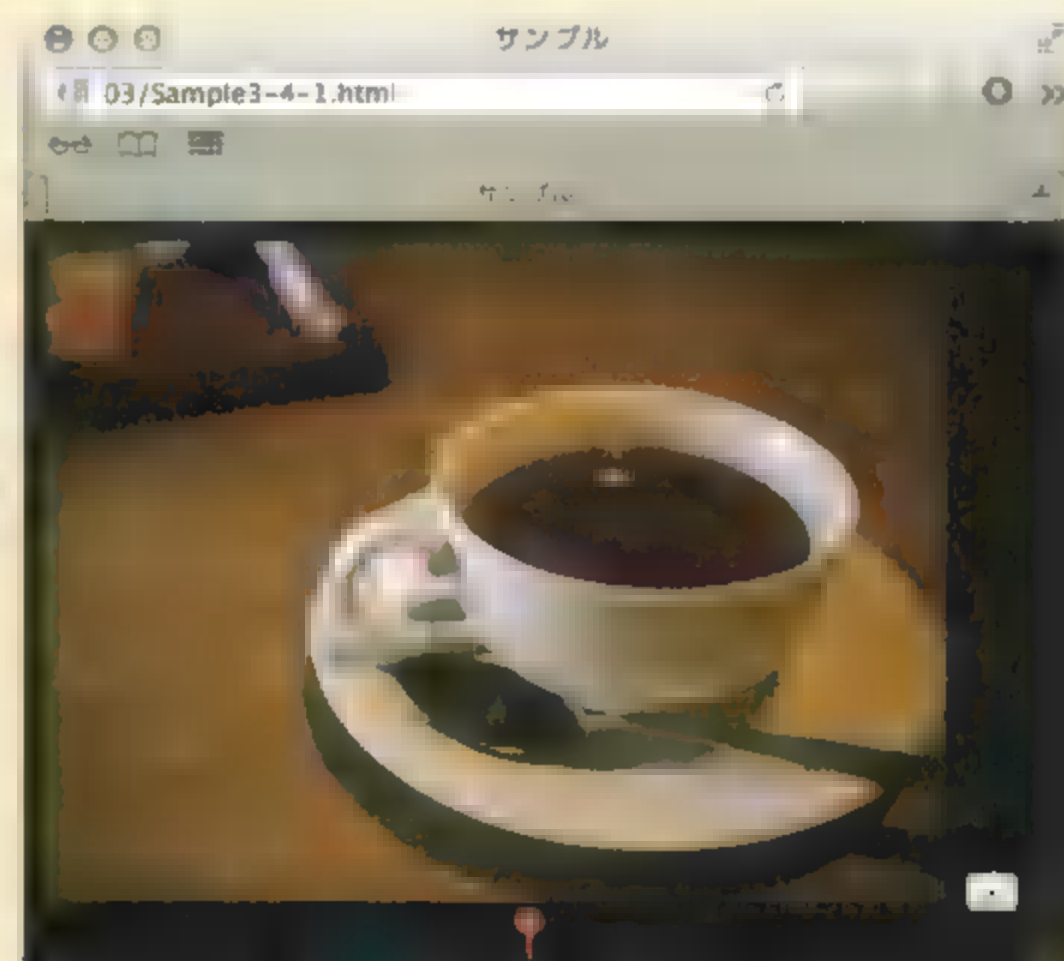
ページの要素を操作するには、次のようなプログラムを作成していきます。

Sample3-4-1.html ■ ページの要素を操作する

```
<!DOCTYPE html>
<html lang="ja">
<head>
<link rel="stylesheet" href="Sample.css">
<title>サンプル</title>
<script type="text/javascript">
```

```
function change()
{
    var e = document.getElementById("image");
    e.setAttribute("src", "pic1.jpg");
}
</script>
</head>
<body>
<img id="image" src="pic0.jpg">
<input type="button" value="●" onclick="change()"/>
</body>
</html>
```

DOMと呼ばれる技術を使用します

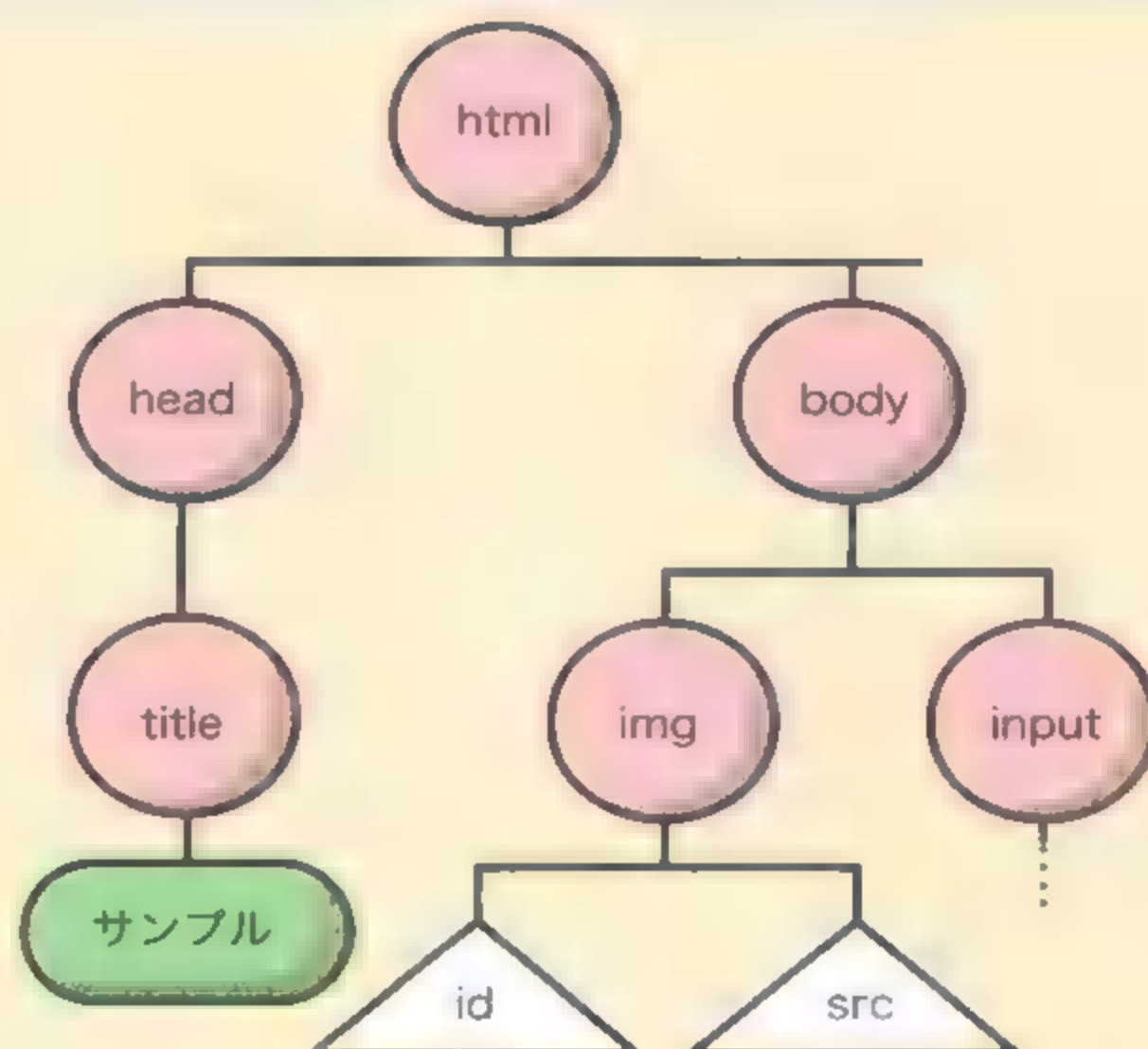
■ Sample3-4-1 の表示

DOM

Webページをきめ細かく操作するには、HTML文書を操作するためのDOM（Document Object Model）と呼ばれる知識が必要となります。DOMは、HTML文書などを扱うためのしくみで、次のようにしてWebページ中の要素を処理します。

ここでは、DOMをHTMLの操作をするために使っています。ただし、DOMはHTML以外にも使われる一般的な技術となっています。HTMLのほか、Webのニュース・記事情報を発信するために利用されているRSS文書を操作するなど、さまざまな場面で利用されています。

要素にID名などをつける

DOMは、HTML文書を木の形をした構造としてとらえたものです。木の節の部分はノードと呼ばれています。各ノードは、JavaScriptではオブジェクト（DOMオブジェクト）として扱われます。

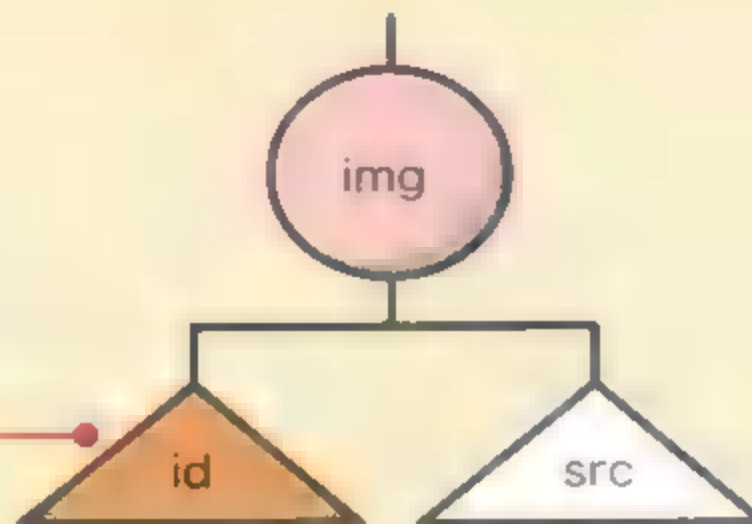
■DOMの構造

DOMのノードはさまざまな種類があります。要素をあらわすオブジェクト、要素にはさまれたテキストをあらわすオブジェクトなどです。いずれもノードとして統一的に取り扱うことができます。

DOMオブジェクトには、さまざまな扱いかたがありますが、ID名を使うと扱いやすくなります。これから操作しようとするHTML要素のタグ中に、id="ID名"という指定を使ってID名を付けておくのです。

そこで、ここでは操作の対象となる「img」要素にimageというID名をつけることにしました。

ID名をつけておきます

```
<img id="image" src="pic0.jpg">
```

■要素にID名をつけておく

DOMオブジェクトを特定する

次に、プログラム中で、この要素をDOMオブジェクトとして特定します。特定するには、DocumentオブジェクトのgetElementById()メソッドによってつけたID名を指定します。

ID名でオブジェクトを指定します

```
var e = document.getElementById("image");
```

これで、変数eによって画像をあらわすDOMオブジェクトを扱うことができます。

オブジェクトを操作する

DOMオブジェクトを取得したら、今度はそれを操作します。ここではオブジェクトの属性を設定する操作を行うことにします。「img」要素のsrc属性を設定して、画像のファイル名を指定することにしましょう。このためにはDOMオブジェクトのsetAttribute()メソッドを使います。

このようにファイル名を指定すれば、画像が差し替えられます。こうしてページの中の構成要素を変更することができるわけです。

オブジェクトを指定します

```
e.setAttribute("src", "pic1.jpg");
```

オブジェクトの操作

DOMでは、Webページやページ中のノードをあらわす、次のようなDOMオブジェクトのメ

ンバを利用して操作を行います。Webページ中のさまざまなノードを操作できるようになっています。

■ DOMの主なメンバ

[illegible]	[illegible]
Documentオブジェクト	
getElementById()	idを指定して要素を取得する
getElementsByTagName()	タグ名を指定して要素リスト取得する
createElement(name)	要素名を指定して要素を作成する
createAttribute(name)	属性名を指定して属性を作成する
createTextNode(text)	テキストを指定してテキストノードを作成する
createComment(text)	テキストを指定してコメントを作成する
Nodeオブジェクト	
appendChild(elem)	指定したノードを子ノードとして追加する
removeChild(elem)	指定した子ノードを削除する
replaceChild(elem1, elem2)	指定した子ノード1を子ノード2で置き換える
innerHTML	ノード内のHTML部分
nodeType	ノード種類 1：要素 2：属性 3：テキスト など
nodeName	ノード名
nodeValue	ノード値
childNodes	子ノードのリスト
firstChild	最初の子ノード
lastChild	最後の子ノード
previousSibling	前の兄弟ノード
nextSibling	次の兄弟ノード
Elementオブジェクト	
createAttribute(name)	指定した属性名の属性を作成する
removeAttribute(name)	指定した属性を削除する
getAttribute(name)	指定した属性名の[illegible]を取得する
setAttribute(name, value)	属性名と属性値を設定する
attributes	属性のリスト

ここでは「img」要素のID属性にID名をつけてDOMで操作し、画像の入れ替えを行う方法を学びました。なお、3-2節では「img」要素のname属性に名前をつけることで画像の入れ替えを行う方法を学んでいます。このように、JavaScriptを使うといくつかの方法で同じ操作を行える場合があります。利用しやすい方法を選び、使いこなせるようになると便利でしょう。

応用！ DOMを操作する

それでは、DOMのメンバを使って、実際にページの操作を行ってみましょう。

テキストを入れ替える

画像のクリック時にテキストを入れ替えることにします。innerHTMLプロパティに文字列を指定すると、テキストを入れ替えることができます。

なお、このサンプルでは、スタイルシートとして前出の「Title.css」を使用します。

Sample3-4-2.html ■ テキストを入れ替える

```
<!DOCTYPE html>
<html lang="ja">
<link rel="stylesheet" href="Title.css">
<head>
<title>サンプル</title>
<script type="text/javascript">
function change()
{
   var e = document.getElementById("content");
   e.innerHTML ="ご利用ありがとうございました。";
}
</script>
</head>
<body>
<h1>Hello</h1>
<p id="content" onclick="change()">お客様のご依頼にお答えいたします。</p>
</body>
</html>
```

入れ替える箇所のDOMオブジェクトを指定します

テキストを入れ替えます

クリック[illegible]に入れ替えることにします

この段落のテキストを入れ替えます

■Sample3-4-2の表示

Tips

innerHTMLプロパティは、指定したDOMオブジェクトの内部に記述されているHTMLデータをあらわしています。ここでは、段落をあらわす「p」要素の中のテキストを意味しているのです。

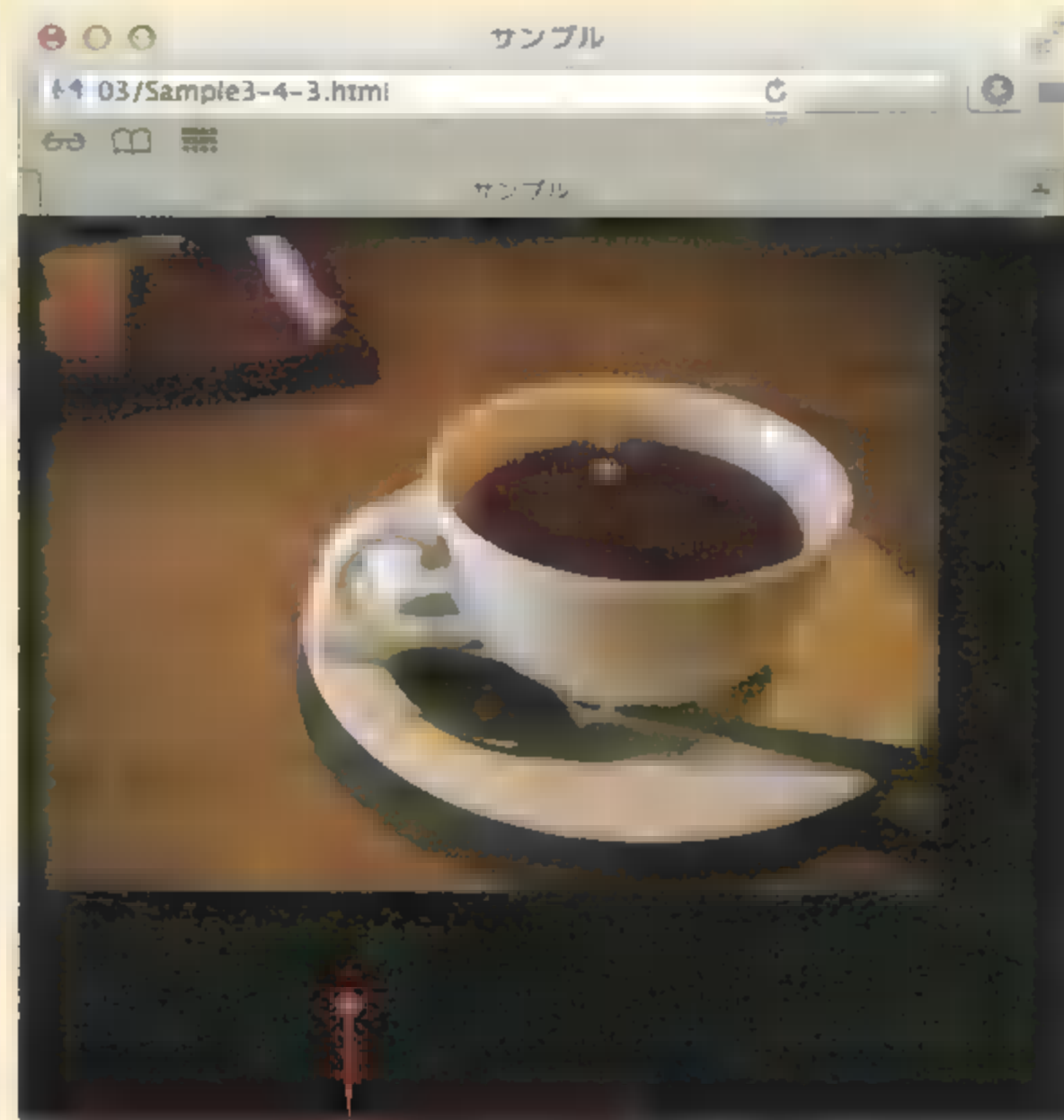
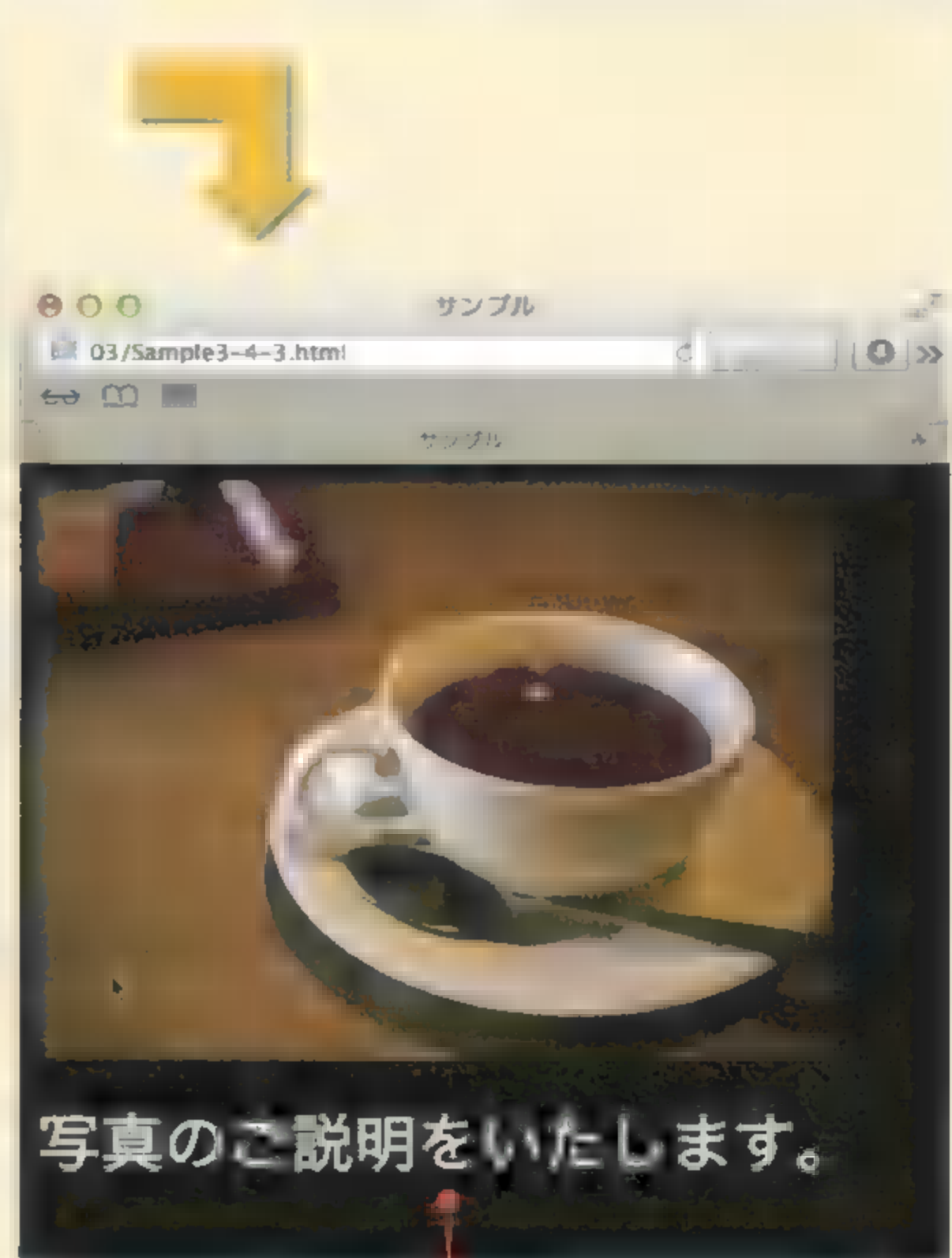
説明を表示する

今度は、画像にマウスをあてたとき（マウスオーバー時）に、操作をしてみることにしましょう。

画像のマウスオーバー時に説明テキストノードを追加し、マウスが離れたら削除します。ノードを追加するには、createTextNode()メソッドでテキストを作成してから、appendChild()メソッドで追加します。削除するにはremoveChild()メソッドを指定します。

Sample3-4-3.html ■ 説明を表示する

```
<!DOCTYPE html>
<html lang="ja">
<link rel="stylesheet" href="Sample.css">
<head>
<title>サンプル</title>
<script type="text/javascript">
function down()
{
   var e = document.getElementById("description");      説明表示部を特定します
   var t = document.createTextNode("写真のご説明をいたします。");
   e.appendChild(t);                                      説明テキストを作成します
}
function up()
{
   var e = document.getElementById("description");      説明表示部を取得します
   e.removeChild(e.lastChild);                            説明テキストを削除します
}
</script>
</head>
<body>
<image id="bigimage" src="pic1.jpg" onmouseover="down()" onmouseout="up()">
<div id="description"></div>          マウスオーバー時に処理します    マウスアウト時に処理します
</body>
</html>
```

■Sample3-4-3の表示

appendChild()には追加するノード、removeChild()では削除するノードを指定します。ここでは、テキストノードを追加し、最後のノード（lastChild）を指定して削除することで、テキストの入れ替えを行っています。

サムネイルから選択する

小さな画像であるサムネイルをクリックすることで、大きな画像を表示するようにしてみましょう。

Sample3-4-4.html ■ サムネイルから選択する

```
<!DOCTYPE html>
<html lang="ja">
<link rel="stylesheet" href="Sample.css">
<head>
<title>サンプル</title>
```

```
<script type="text/javascript">
function change(picname)
{
   var e =  document.getElementById("bigimage");
   e.setAttribute("src", picname);
}
</script>
</head>
<body>
<img id="bigimage" src="pic0.jpg">
<img id="smallimage" src="pic0.jpg" onclick="change('pic0.jpg')">
<img id="smallimage" src="pic1.jpg" onclick="change('pic1.jpg')">
<img id="smallimage" src="pic2.jpg" onclick="change('pic2.jpg')">
</body>
</html>
```

大きな画像を取得します

クリックしたサムネイル画像を表示します

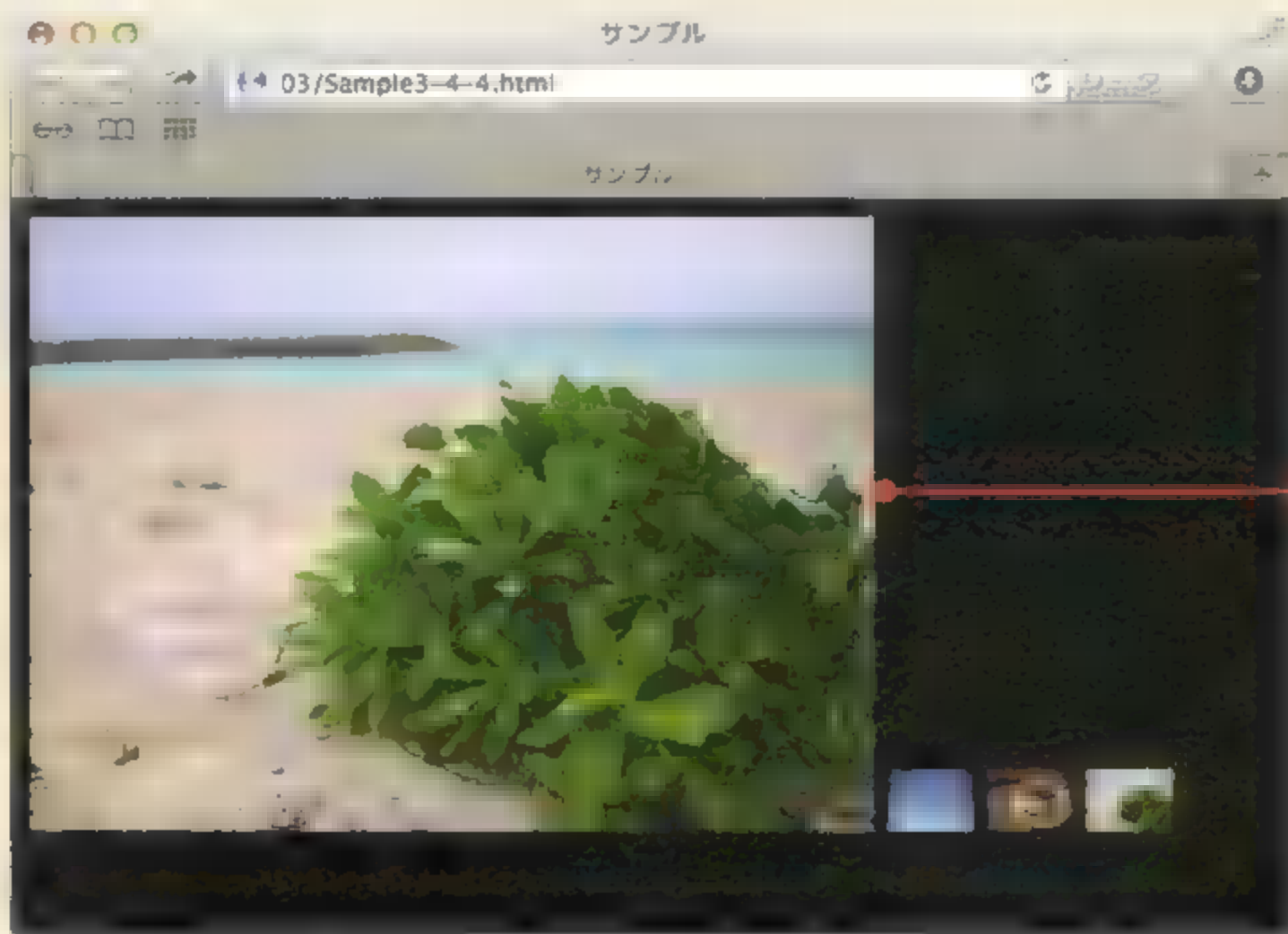
大きな画像を表示します

サムネイルを表示します

サムネイル画像のファイル名を渡します

大きな画像をあらわす「img」要素にbigimage、小さなサムネイル画像をあらわす「img」要素にsmallimageというID名をつけました。

ここでは、サムネイル画像をクリックしたときに、大きな画像（bigimage）にサムネイル画像のファイル名（pic0.jpg〜.pic2.jpg）を引き渡して表示するしくみになっています。

■ Sample3-4-4の表示

引数 Column

　ここでは、イベント時の処理をする際に、画像名の情報を関数に渡しています。これをプログラミングの世界では引数といいます。ここでは、change()処理をする際に、クリックした画像名pic0.jpgを渡しています。

```
<img id="smallimage" src="pic0.jpg" onclick="change('pic0.jpg')">
```

画像名を引数として指定しています

　渡される処理の側では、変数を使って情報を記憶することになります。たとえば、ここでは変数picnameに記憶します。
　処理の中では渡された情報を扱うことができます。ここでは、picnameで取り扱うことができるのです。
　つまり、こんなふうに情報が渡され、処理されているのです。

画像名を引数として渡します

```
<img id="smallimage" src="pic0.jpg" onclick="change('pic0.jpg')">
```

```
function change(picname)
{
   var e =  document.getElementById("bigimage");
   e.setAttribute("src", picname);
・・・
```

渡された情報（画像名）を記憶します

渡された情報（画像名）を利用します

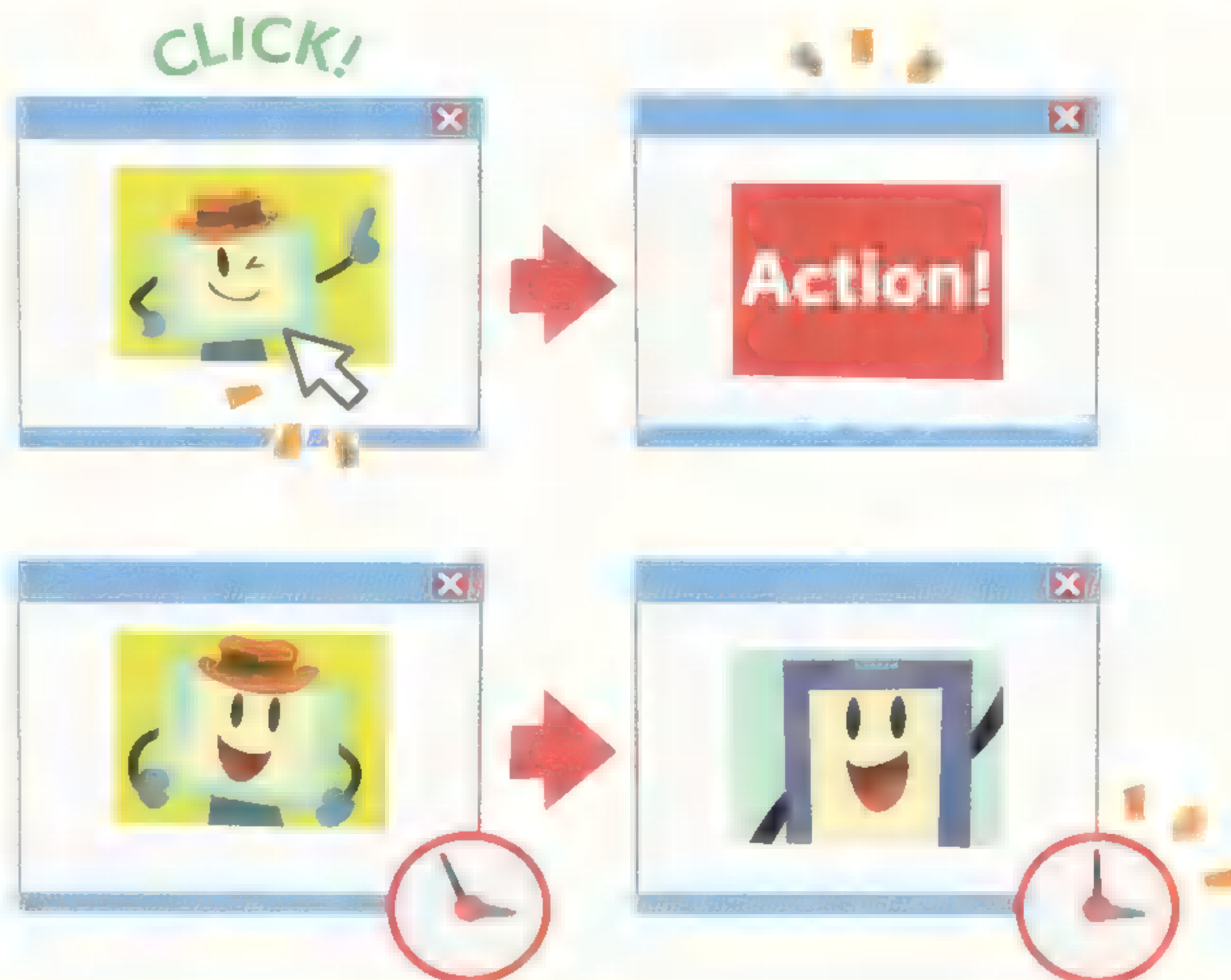
3-5 タイマーを使おう！

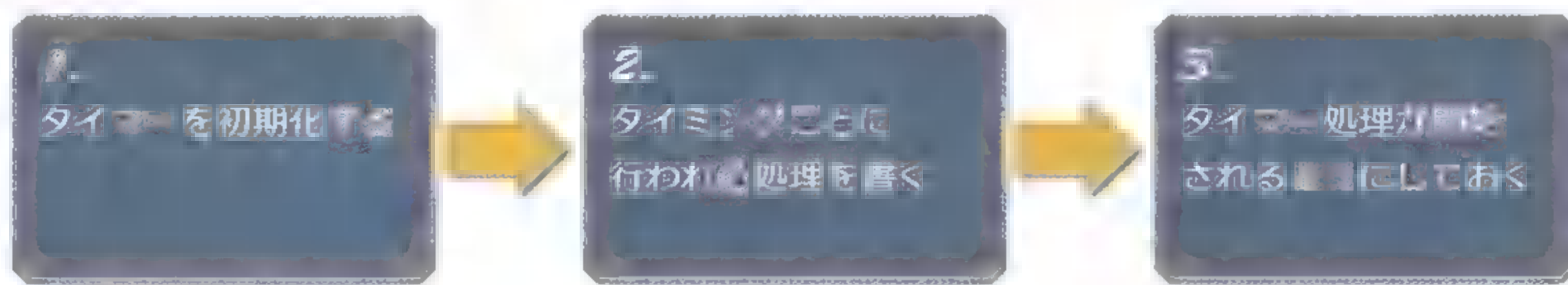
チャレンジ21 自動的に処理しよう

さて、これまでは、さまざまなタイミングでページの操作をしてきました。ユーザーがダブルクリックでWebページを表示したり、ボタンをクリックするなど、手動で操作をしたというイベントが発生したときに処理をしたわけです。

これに対して、自動的にWebページが動作するようにする方法があります。つまり、JavaScriptを使って、自動的に発生するイベントによって処理が行われるようにするのです。

自動的にイベントを発生させるJavaScriptの機能をタイマーと呼びます。タイマーを使って自動的に処理をしてみることにしましょう。

■ユーザーの操作によるイベント（上）、自動で発生するイベント（下）

タイマーを使うには？

タイマーは次のように使います。順番にみていきましょう。

タイマーを初期化する

まず、タイマーの初期設定を行います。初期化処理を適当な処理名でまとめておきましょう。ここではinit()という名前にします。

この処理の中で、タイマーが時を刻むように設定することにします。このためには、Windowオブジェクトのsetinterval()メソッドを使います。

```
function init()
{
   window.setInterval("tick()", 3000);   ← 3秒ごとのタイミングを指定します
}
```

タイマーによって発生するイベントを処理する処理名は、tick()とすることにしましょう。この処理名は自分で決めることができますので、上のように" "で囲って指定してください。

また、処理が行われる間隔をミリ秒で指定します。ここでは、3秒（3000ミリ秒）と指定しているわけです。

タイマー処理を書く

それでは、決めた処理名の中にタイマー処理をまとめましょう。3秒ごとに行う処理を書くのです。ここでは、次の画像に入れ替える処理をしています。番号が7だった場合は0に戻しています。画像を次々と表示する方法を思い出してみましょう。

```
function tick()
{
   i++;
   if(i == 7) i=0;
   var e = document.getElementById("image");
   e.setAttribute("src", "pic" + i +".jpg");   ← 画像を入れ替えます
}
```

タイマーを開始する

　最後に、タイマーが開始されるようにしておきます。ここでは、ページの本体である「body」要素を読み込んだときに（onload)、初期化処理を指定することにします。これでタイマーが開始されることになります。

　実際にページが読み込まれ、タイマーが開始されると、初期設定で指定したとおりに、3秒ごとにイベントが発生して処理が行われるというわけです。

```
</script>
</head>
<body  onload="init()">
```

ページを読み込んだタイミングでタイマー処理を開始します

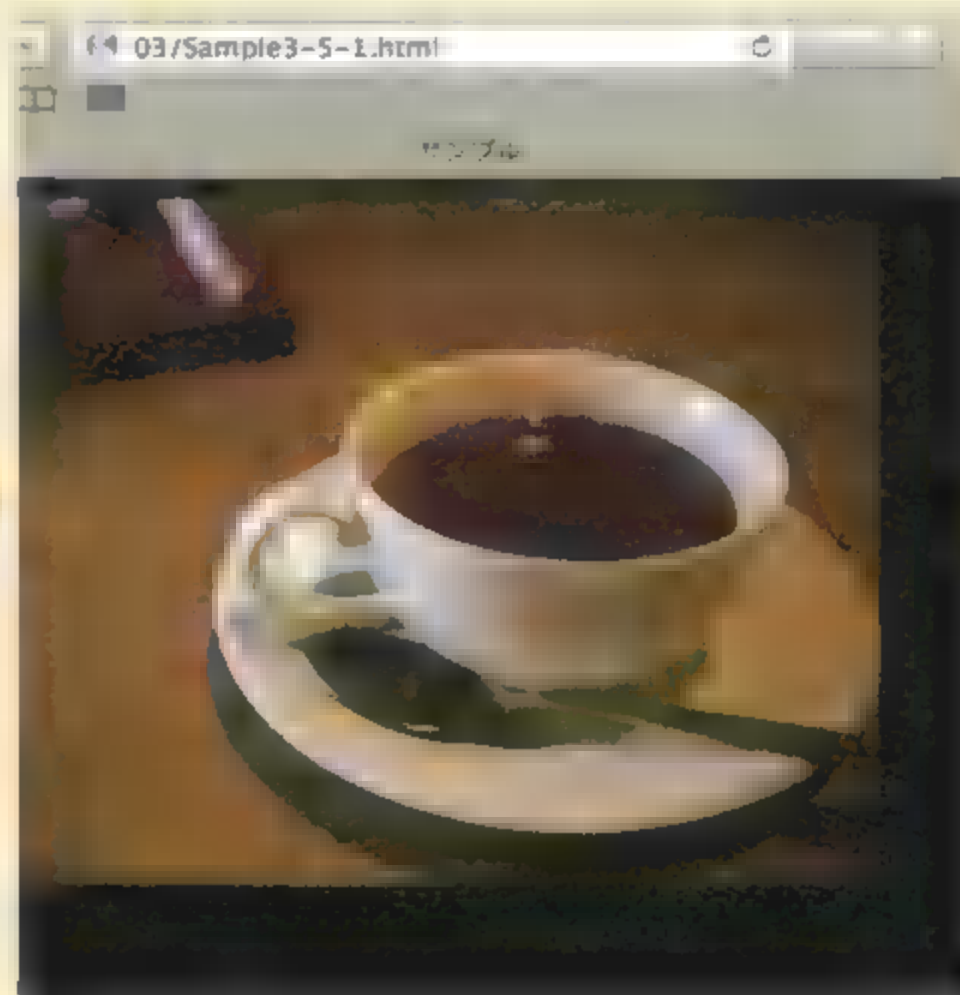
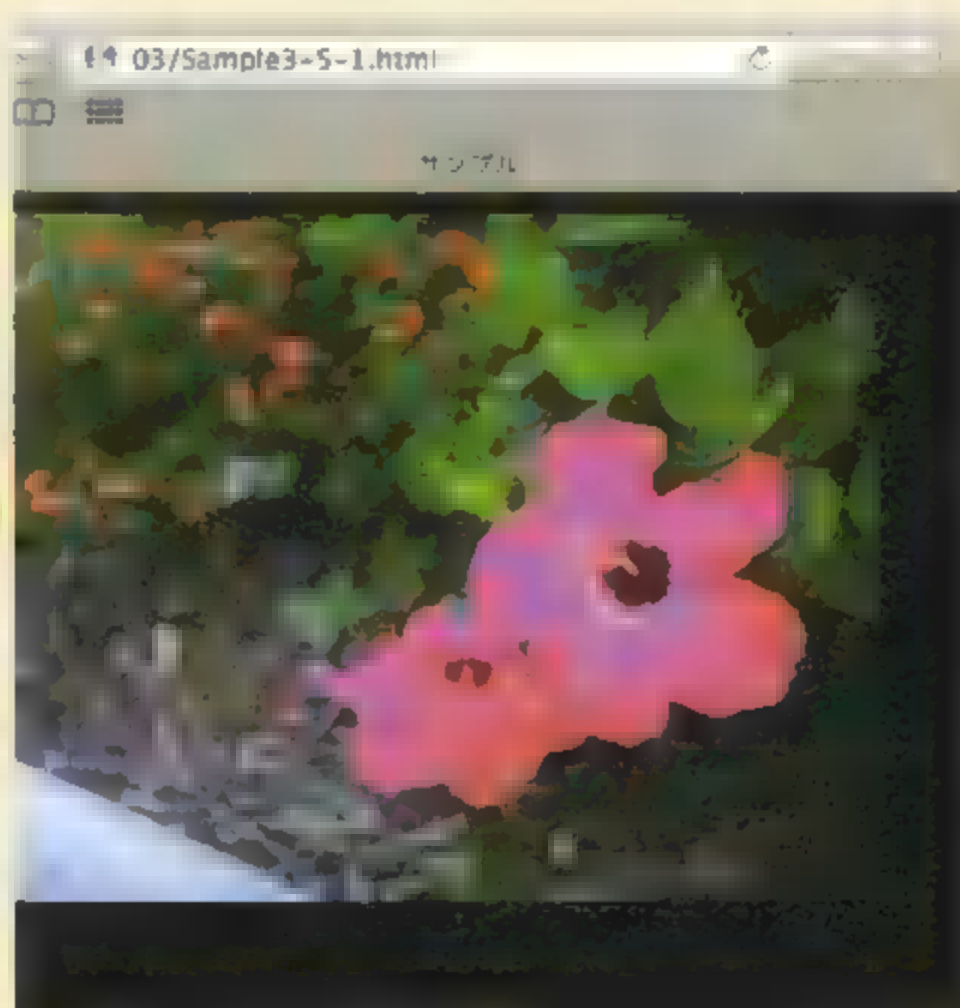
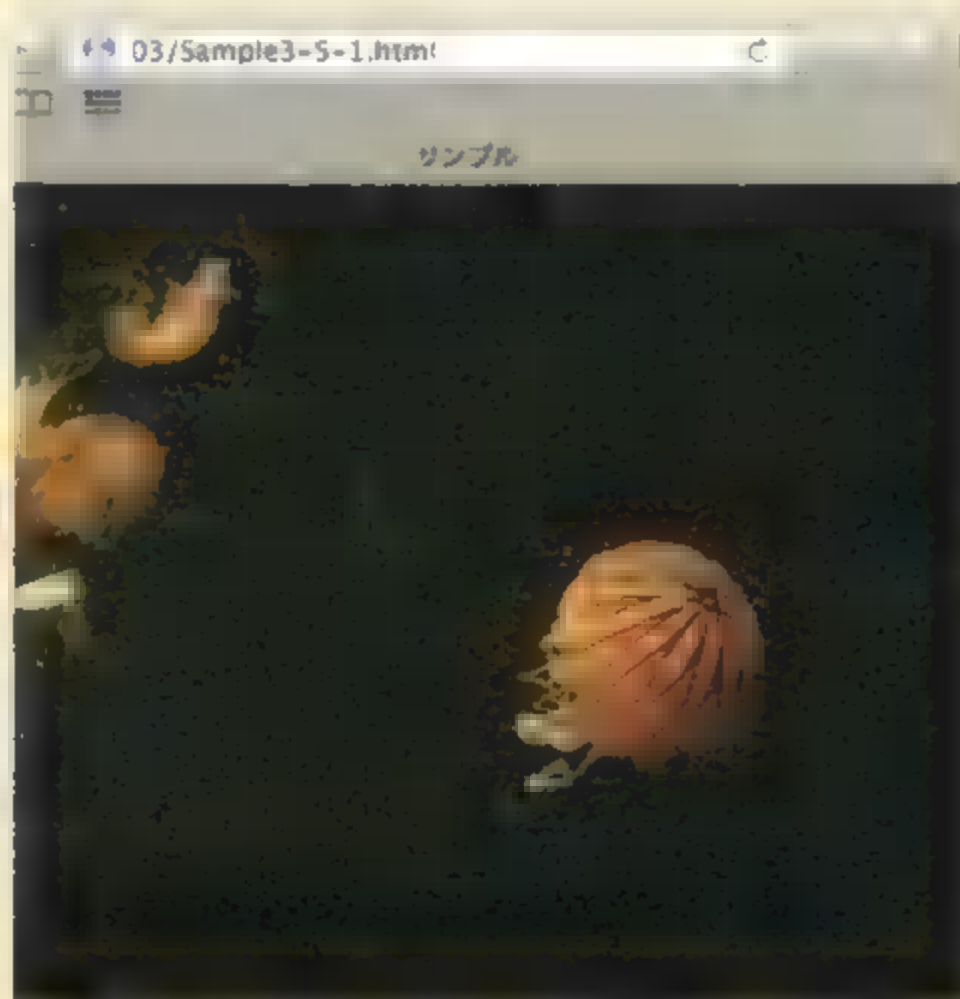
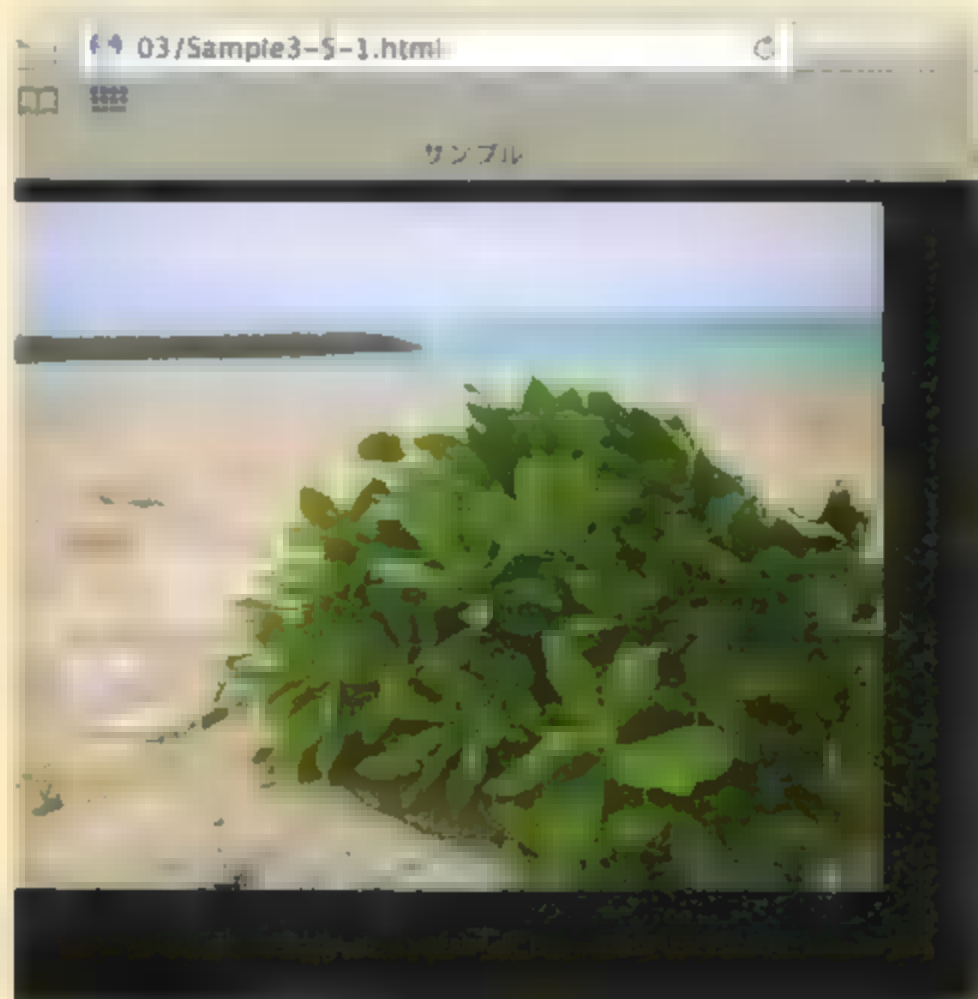
タイマーで画像を入れ替える

　それでは、この手順にしたがってタイマーを使い、3秒ごとのタイミングで画像を入れ替えてみましょう。自動的に入れ替わる画像を確認してみてください。

Sample3-5-1.html ■ タイマーを使う

```
<!DOCTYPE html>
<html lang="ja">
<head>
<link rel="stylesheet" href="Sample.css">
<title>サンプル</title>
<script type="text/javascript">
var i=0;
function init()
{
   window.setInterval("tick()", 3000);
}
function tick()
{
   i++;
   if(i == 7) i=0;
   var e = document.getElementById("image");
   e.setAttribute("src", "pic" + i +".jpg");
}
</script>
</head>
<body onload="init()">
<img id="image" src="pic0.jpg">
</body>
</html>
```

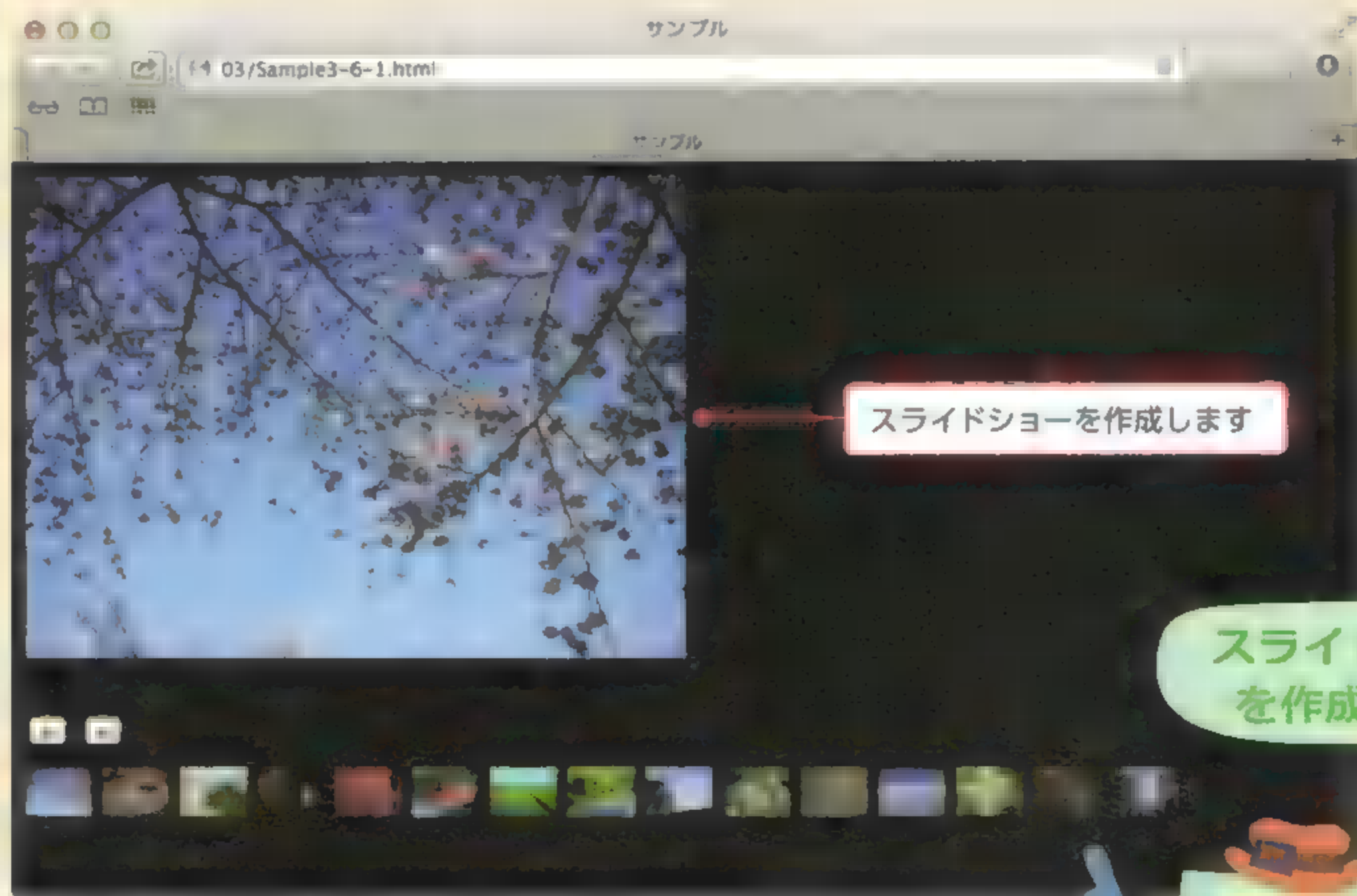
- function init() — タイマーの初期化処理を行います
- window.setInterval("tick()", 3000); — タイマーを設定します
- var e = document.getElementById("image"); — ID名でオブジェクトを指定します
- e.setAttribute("src", "pic" + i +".jpg"); — 画像を入れ替えます
- <body onload="init()"> — ページを読み込んだタイミングでタイマー処理を始めます

■Sample3-5-1の表示

3-6 スライドショーにしよう

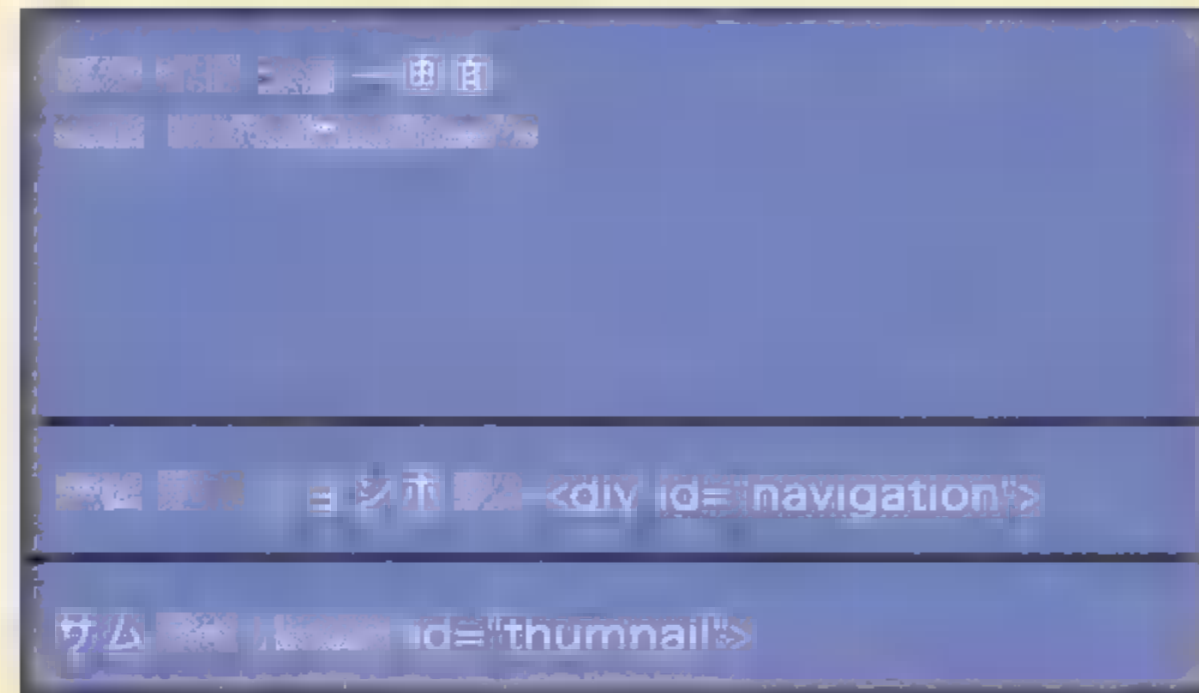
チャレンジ! スライドショーにする

ここまでページ中の要素を入れ替えたり、タイマーを使って自動的に処理をする方法を紹介しました。そこでこの章の最後に、これまでに紹介した方法を使って、Webページ上にスライドショーを作成してみましょう。

次のように、大きなスライドショー画面と小さなサムネイル画像から構成するスライドショーとします。サムネイルに表示されている画像を、タイマーで順番に表示します。タイマー操作を行うボタンもつけることにしましょう。

■スライドショー

Step 1 画面全体を設計する

まず、スライドショーの画面を設計しましょう。スライドショーのメイン画面・ナビゲーションボタン・サムネイル画面に分割して考えます。このようにWebページを複数の範囲にわける場合には、HTMLの「div」要素のタグを使い、各範囲にID名をつけておく方法がよく利用されます。

■div要素で画面を分割して考える

Step 2 サムネイル部分を設計する

サムネイル部分には、15枚の画像を表示することにします。画像の数が多くなるため、JavaScriptの繰り返し文を使ってサムネイル画像を表示することにしましょう。15行の「img」要素を書かなくても、繰り返し文を使えば、簡単に表示することができます。多くのサムネイル画像を簡単に表示する方法を思い出してみましょう。

```
for(var i=0; i<15; i++){
  document.writeln('<img class="smallimage" src="pic' + i +'.jpg">');
}
```

サムネイルを繰り返し表示します

なお、サムネイルは、画像を縮小して表示することにします。そこで、画像の大きさをスタイルシート中に指定することにしましょう。

```
.smallimage{
  width: 40px;
  height: 30px;
}
```

画像を縮小表示します

Step3 ボタンの動作を設計する

次に動作の設計です。スタートボタン・ストップボタンの動作を作成することにします。スタートボタンを押したときにタイマー処理を開始し、ストップボタンを押したときにタイマー処理を終了します。各ボタンのイベントと処理名を指定してください。

```
var k=0;
var t=null;
function stat()
{
   if(t == null)
      t = window.setInterval("tick()", 3000);
}
function stop()
{
   window.clearInterval(t);
   t = null;
}

...

<input type="button" value="●" onclick="stat()"/>
<input type="button" value="■" onclick="stop()"/>
```

スタートボタンを押したときの処理です

ストップボタンを押したときの処理です

スタート時の処理を行います

ストップ時の処理を行います

WindowオブジェクトのclearInterval()メソッドを使うことで、タイマーをストップすることができます。このためにはタイマー開始時に戻されるオブジェクトを変数(ここではt)に記憶しておき、この変数を指定してストップします。varを使った変数tの準備は各処理の外側に書くようにしてください。

Step4 タイマーを設計する

それでは、メインとなるタイマーの処理を設計しましょう。変数kが15であったときに0に戻ることにします。

```
function tick()
{
   k++;
   if(k == 15) k=0;
```

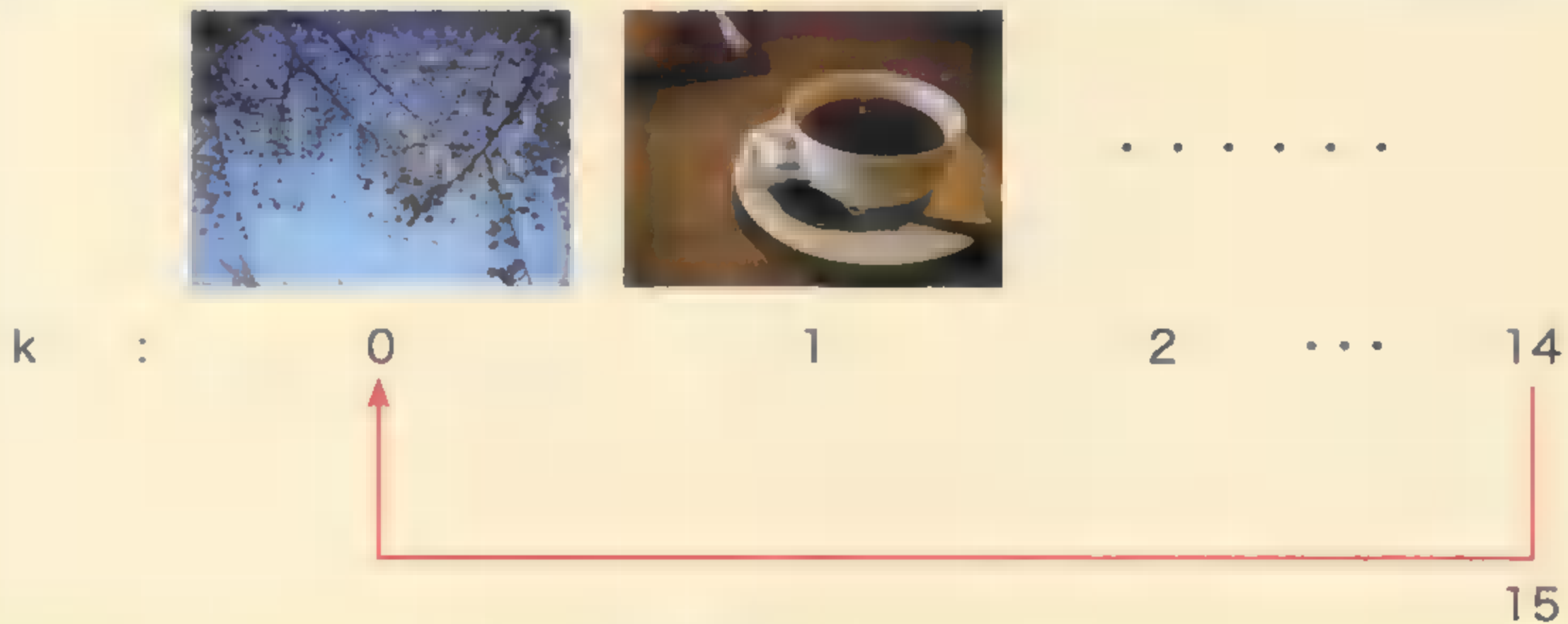
次の画像番号を指定します

[illegible]番号が最後の番号をこえたら最初に戻ります

```
    var e = document.getElementById("bigimage");
    e.setAttribute("src", "pic" + k +".jpg");
}
```

大きい[illegible]あらわす[illegible]を指定し・・・

画像を入れ替えます

■画像を次々と表示する

Step5 スライドショーを完成させよう

最後に、プログラムを完成させることにしましょう。完成したスライドショーのボタンを操作し、正しく動作しているか確認してみてください。3秒ごとにスライドショーが切り替われば完成です。

Sample3-6-1.html ■ スライドショー

```
<!DOCTYPE html>
<html lang="ja">
<head>
<link rel="stylesheet" href="SlideShow.css">
<title>サンプル</title>
<script type="text/javascript">
var k=0;
var t=null;
function stat()
{
    if(t == null)
        t = window.setInterval("tick()", 3000);
}
function stop()
```

スタートボタンを押したときの処理です

ストップボタンを押したときの処理です

```
{
    window.clearInterval(t);
    t = null;
}
function tick()
{
    k++;
    if(k == 15) k=0;
    var e = document.getElementById("bigimage");
    e.setAttribute("src", "pic" + k +".jpg");
}
</script>
</head>
<body>

<div id="slideshow">
<img id="bigimage" src="pic0.jpg">
</div>

<div id="navigation">
<input type="button" value="●" onclick="stat()"/>
<input type="button" value="■" onclick="stop()"/>
</div>

<div id="thumbnail">

<script>
for(var i=0; i<15; i++){
    document.writeln('<img class="smallimage" src="pic' + i +'.jpg">');
}
</script>

</div>

</body>
</html>
```

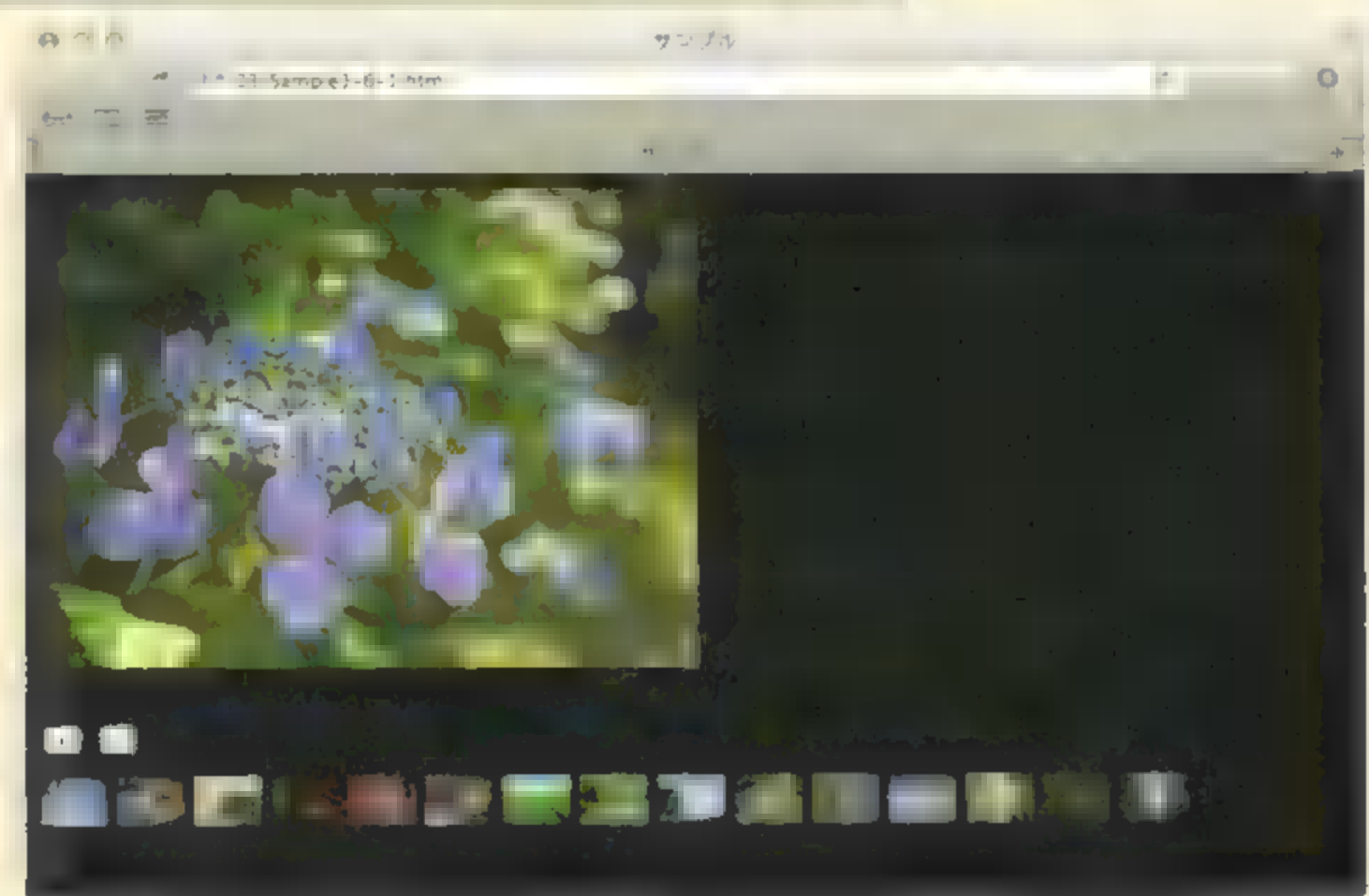
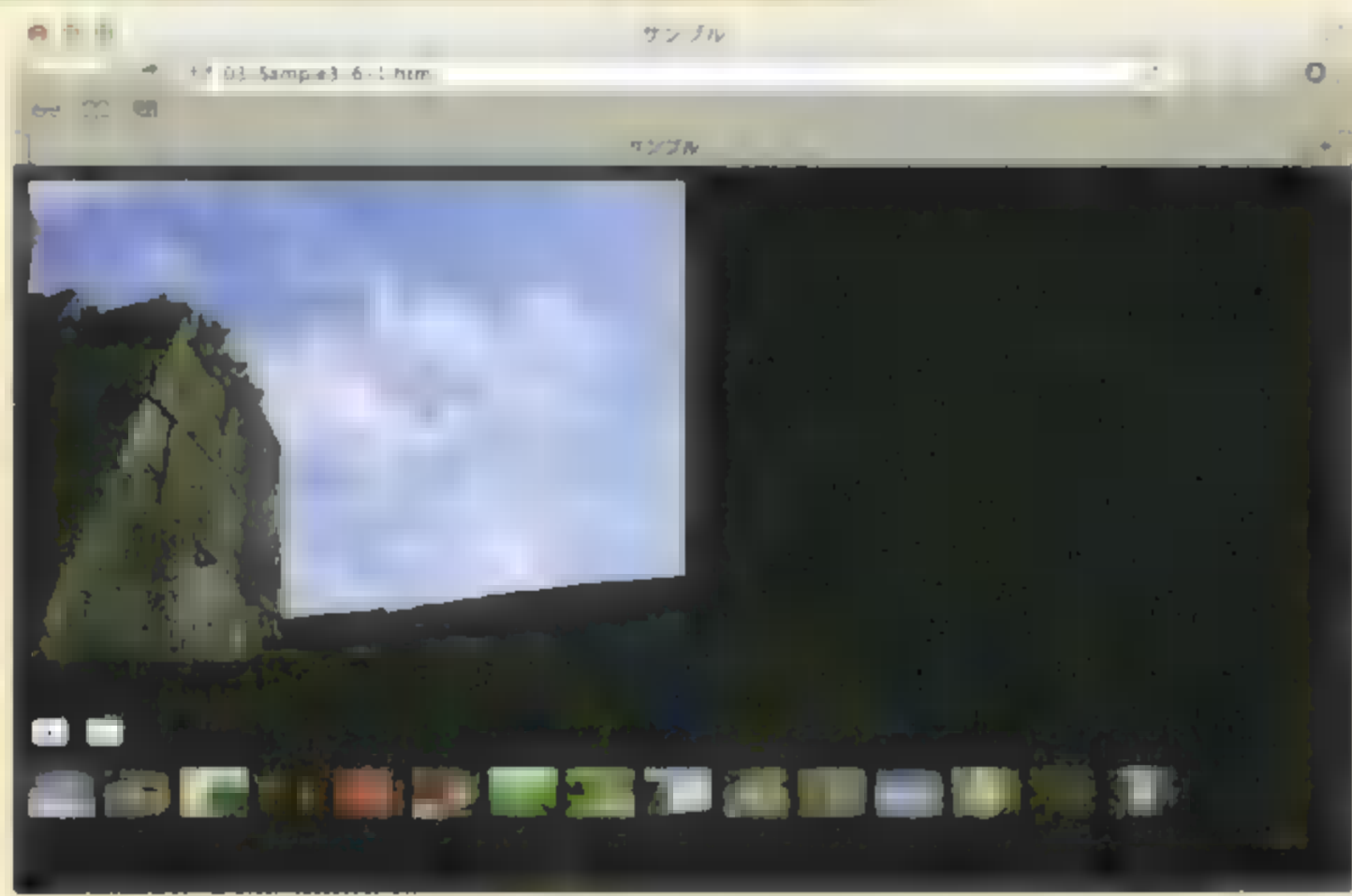
- 次の画像番号を指定します（k++;）
- [illegible]番号が最後の番号をこえたら最初に戻ります（if(k == 15) k=0;）
- 大きい画像をあらわす要素を指定し・・・（var e = document.getElementById("bigimage");）
- 画像を入れ替えます（e.setAttribute(...)）
- スライドショー画像を表示します（<img id="bigimage" src="pic0.jpg">）
- スタート[illegible]の処理を行います（onclick="stat()"）
- ストップ[illegible]の処理を行います（onclick="stop()"）
- サムネイルを繰り返し表示します（for(var i=0; i<15; i++){）

SlideShow.css ■ スライドショーのスタイルシート

```
body{
    background-color: #000000;
    color: #FFFFFF;
    font-size: 2em;
    font-family: fantasy;
}
```

```
.smallimage{
   width: 40px;
   height: 30px;
}
```

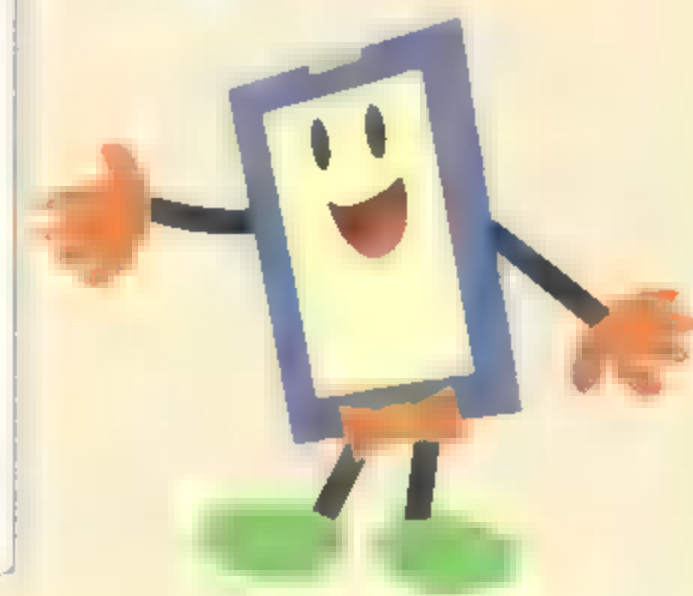
画像を縮小表示します

■ Sample3-6-1 の表示

4-1 入力フォームを作ろう

チャレンジ！ 入力できるようにしよう

Webページには、ユーザーがデータを入力するための部品を配置することができます。この部品を配置するためのHTML要素をフォームと呼びます。

JavaScriptを使えば、フォームを活用していくことができます。この節ではフォームを活用してみることにしましょう。

■フォーム

フォーム部品の種類

フォームには、ユーザーがキーボードやマウスを使ってデータを入力するための部品を配置します。フォーム部品には、次のような種類があります。

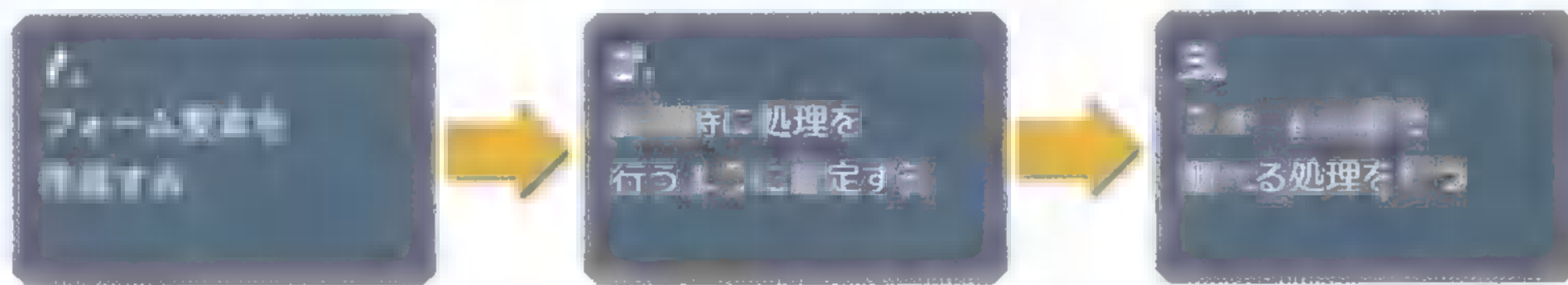
■ フォーム部品

種類	[illegible]	[illegible]
フォーム		<form name="名前" action="URL" method="種類">
テキストボックス		<input type="text" name="名前" value="値" />
テキストエリア	鉛筆やコンパスを注文します	<textarea name="名前" cols="1行文字数" rows="行数" ></textarea>
チェックボックス	☑コンパス □ボールペン	<input type="checkbox" name="名前" value="値" >表示</input>
ラジオボタン	○コンパス ◉ボールペン	<input type="radio" name="名前" value="値" >表示</input>
ドロップダウンメニュー	✓東京 名古屋 京都 大阪 福岡	<select name="名前"> <option value="値">表示</option> </select>
リストボックスメニュー	鉛筆 消しゴム 定規 コンパス ボールペン	<select name="名前" size="値"> <option value="値">表示</option> </select>
ボタン	表示	<input type="button" value="値" />
送信	送信	<input type="submit" value="値" />
リセット	リセット	<input type="reset" value="値" />
ファイル	ファイルを選択	<input type="file" name="名前"/>
隠し属性		<input type="hidden" name="名前" value="値"/>

Tips

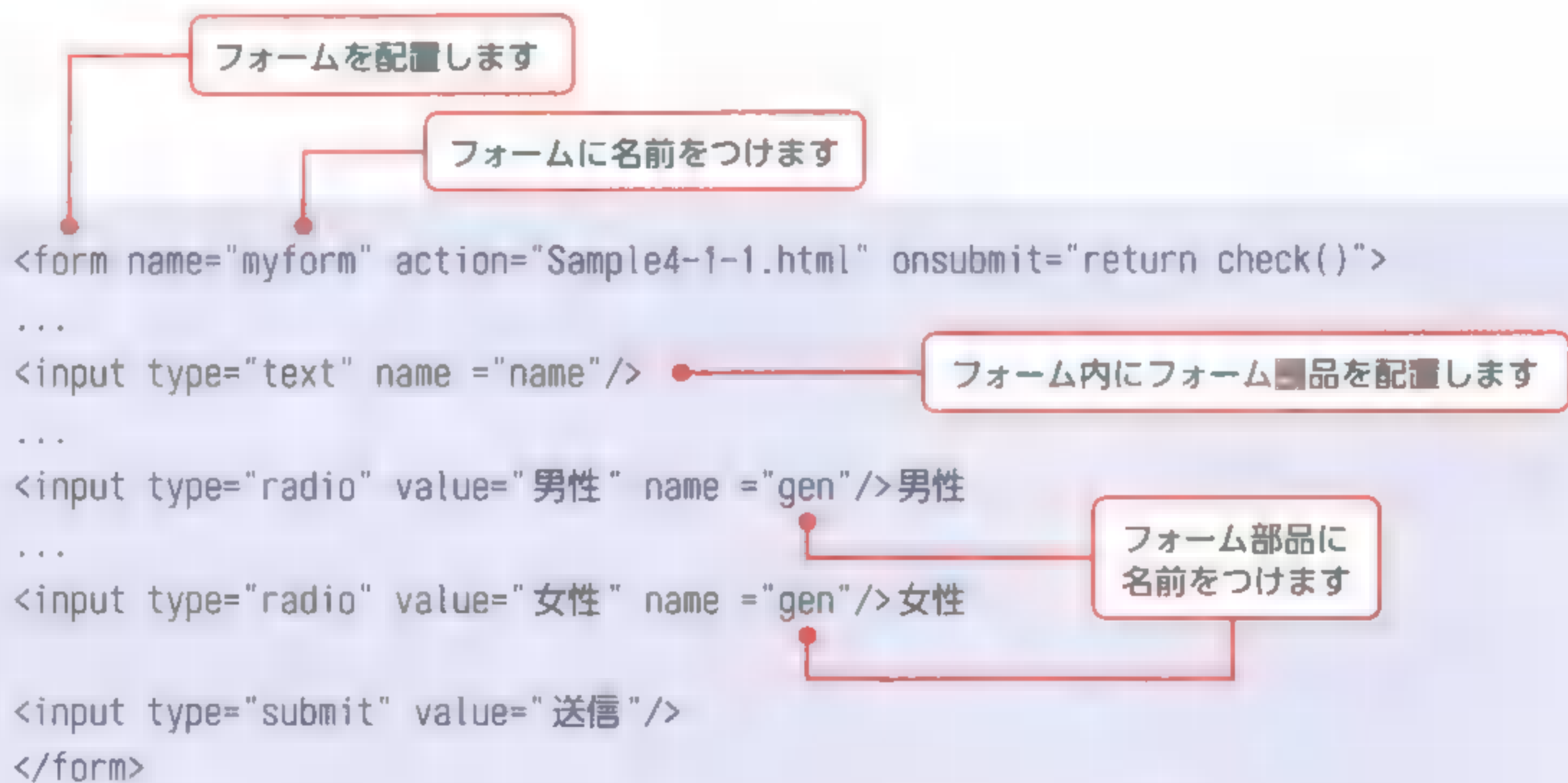
これまでの章でも、フォーム部品の一種であるボタンを使ってきました。フォーム部品は、単独でも使えますが、Webサーバーにデータを送信するためには、部品をフォーム要素の中に配置することが必要です。この章ではフォーム要素の中に部品を配置してデータを送信することにします。

フォームを使うには？

それでは、フォームの使い方についてみておきましょう。フォームは、一般的に次のような手順で使います。

フォーム要素を作成する

フォームは、HTMLの「form」要素を使ってあらわします。フォーム部品はフォームの中に配置します。「form」要素の中に書くわけです。

通常、送信データを特定するために、フォームとその部品には、name属性を使って名前をつけておきます。名前はわかりやすいものをつけてかまいません。

たとえば、下の場合はフォームにmyformという名前をつけ、テキストボックスにname、ラジオボタンにgenという名前をつけています。

フォーム送信時に設定処理をする

フォームは、部品に入力したデータを送信する際に処理を行うことが多いため、通常、部品をあらわすタグに、onsubmit="処理名()"と指定して処理を行うように指定します。ここでは、check()処理を行うことを指定してみました。

```
<form name="myform" action="Sample4-1-1.html" onsubmit="check()">
```

処理を指定します

actionにはデータの送信先を指定します。通常は、データを受け取るWebサーバー上のプログラムのURLを記述しますが，本書では送信元と同じ場所（Sample4-X-X.html）を指定することにします。

フォームのデータを調べる

このようにしておくと、送信時に行われる処理の中で、フォーム名.フォーム部品名.valueという指定を使い、フォームで送信されるデータを調べることができます。たとえば、myformフォーム内のテキストボックスnameの値を変数nに記憶するには、次のように指定します。

```
...
function check()
{
   var n = myform.name.value;
   ...
```

フォームとその部品の名前から値を調べることができます

入力されたデータを調べる

それでは、フォームに入力されたデータを調べてみましょう。送信ボタンを押したときに、フォームのデータをウインドウに表示することにします。なお、この章では次のスタイルシートを使います。

Form.css ■ フォーム用のスタイルシート

```
body{
   background-color: #FFFFFF;
   color: #000000;
   font-family: fantasy;
}
table{
   background-color: #66cdaa;
   border: #FFFFFF 2px solid;

}
```

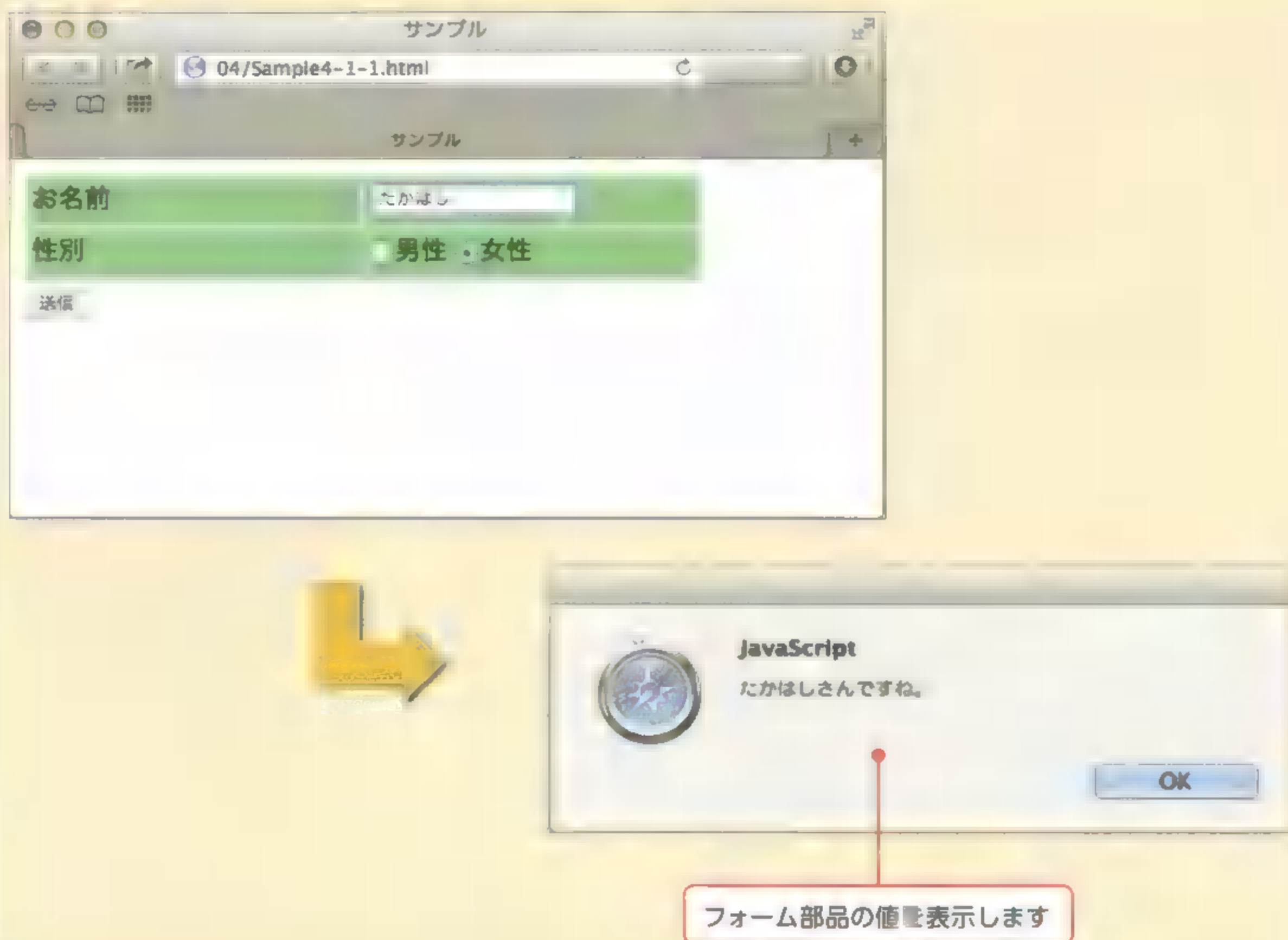
4 JavaScriptでチェックしよう！

```
td{
    width: 200px;
    border: #FFFFFF 1px solid;
}
```

Sample4-1-1.htm ■ フォームの入力を調べる

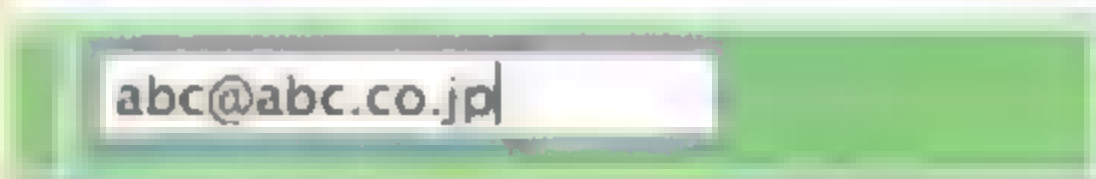
```
<!DOCTYPE html>
<html lang="ja">
<head>
<link rel="stylesheet" href="Form.css">
<title>サンプル</title>
<script type="text/javascript">
function check()
{
    var n = myform.name.value;            ← フォーム部品の値を調べます

    window.alert(n + "さんですね。");     ← フォーム部品の値を表示します
}
</script>
</head>
<body>
<form name="myform" action="Sample4-1-1.html" onsubmit="check()">
<table>
<tr>
<td>お名前</td>
<td>
<input type="text" name ="name"/>
</td>
</tr>
<tr>
<td>性別</td>
<td>
<input type="radio" value="男性" name ="gen" checked/>男性
<input type="radio" value="女性" name ="gen"/>女性
</td>
</tr>
</table>
<input type="submit" value="送信"/>
</form>
</body>
</html>
```

■ Sample4-1-1の表示

フォーム部品の値を表示します

フォームの送信

　ここではJavaScriptを使って、送信するデータをWebクライアント上で調べるしくみになっています。Webページをみているユーザーの側でチェックを行っているわけです。

　しかし、Webサイトを実際に設計する場合には、送信データをWebサーバー上で調べることが必要となるでしょう。実際に送信されたデータをWebサーバー上で調べるには、Webサーバー上で動作し、データなどをチェックするプログラミング環境が必要です。

　Webサーバー上で動作するプログラミング環境・言語としては、PHPなどが使われています。JavaScriptは、ユーザー側の環境でチェックを行うものです。このためJavaScriptは、送信時に簡単なチェックをする目的で使われることが多くなっていることに注意してください。

4-2 入力をチェックしよう

チャレンジ!! フォームのチェック機能を充実させる

JavaScriptを使って、フォームの送信時にフォームに入力したデータの値を調べることができました。

実際には、このフォームの入力値についていろいろなチェックを行ってから、Webサーバーにデータを送信する機会が多くなっています。そこで、今度はフォームの値をチェックして送信処理を行うようにしましょう。

■ 送信時にチェックする

abc@abc.co.jp

入力をチェックする

まず、フォームに値が入力されているかをチェックしてみましょう。値が空になっている場合に、「入力してください。」と表示するようにしてみます。入力されていた場合には、メッセージを表示して送信します。

Sample4-2-1.html ■ 入力をチェックする

```
<!DOCTYPE html>
<html lang="ja">
<head>
<link rel="stylesheet" href="Form.css">
<title>サンプル</title>
<script type="text/javascript">
```

```
function check()
{
    var n = myform.name.value;

    if(n == ""){
        window.alert("入力してください。");
        res = false;
    }
    else{
        window.alert("送信します。");
        res = true;
    }
    return res;
}
</script>
</head>
<body>
<form name="myform" action="Sample4-2-1.html" onsubmit="return check()">
<table>
<tr>
<td>お名前</td>
<td>
<input type="text" name ="name"/>
</td>
</tr>
<tr>
<td>性別</td>
<td>
<input type="radio" value="男性" name ="gen" checked/>男性
<input type="radio" value="女性" name ="gen"/>女性
</td>
</tr>
</table>
<input type="submit" value="送信"/>
</form>
</body>
</html>
```

フォームの入力が空だったときには・・・

入力してもらうようにして・・・

送信をキャンセルします

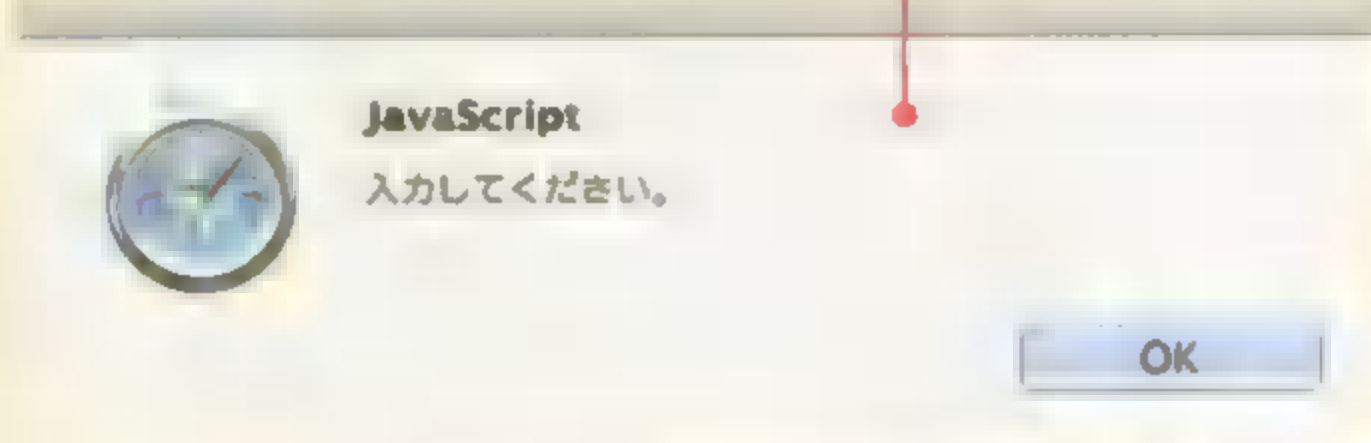
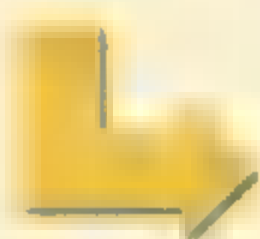
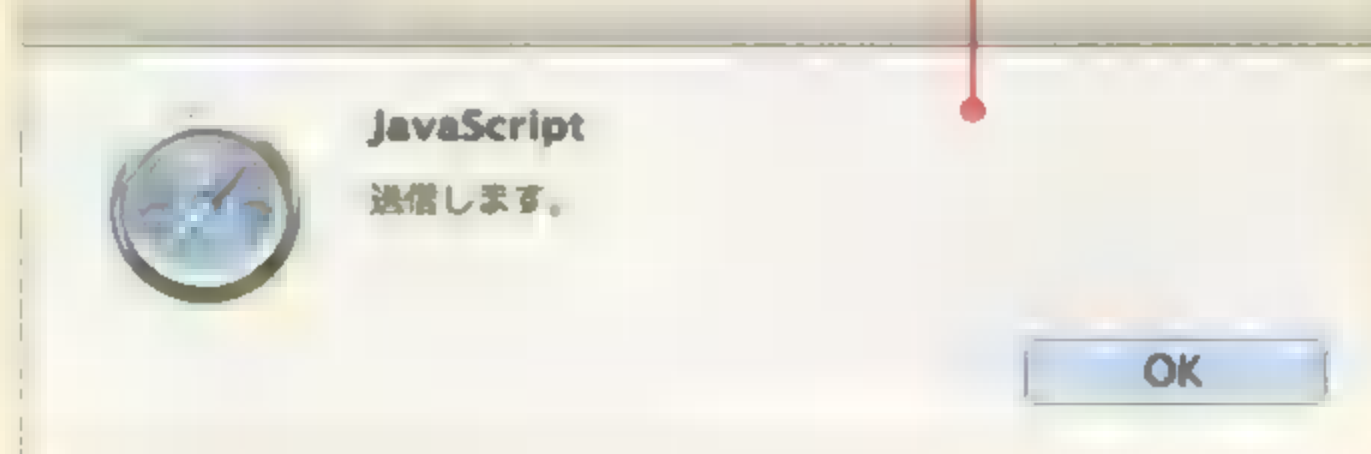
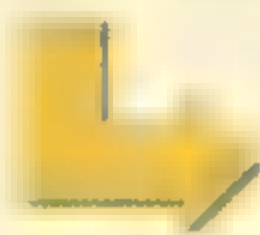
入力されていた場合には・・・

メッセージを表示し・・・

送信します

処理結果を調べる場合に指定します

フォームに入力しなかった場合と、入力した場合のそれぞれについて確認してみてください。

■ Sample4-2-1の表示

フォームへの入力を確認するには？

ここでは、if文を使って入力されていなかった場合にだけ、送信をキャンセルしています。フォームに入力されていた場合には送信しています。場合分けをしているのです。

```
if(n == ""){
    window.alert("入力してください。");
    res = false;
}
else{
    window.alert("送信します。");
    res = true;
}
return res;
```

- `if(n == ""){` ― 空だったときには・・・
- `res = false;` ― 入力してもらうようにします
- `else{` ― そうでない場合には・・・
- `res = true;` ― 送信します

onsubmitでは、指定した処理が「true（OK）」という結果を戻したとき、送信が行われます。「false（OKではない）」を戻した場合には送信は行われません。ここでは、このことを利用して送信・キャンセル処理を行っています。returnは結果を戻すための指定です。onsubmit="return 処理名()"と指定しておきます。

応用！ 入力チェックのバリエーション

フォームの入力チェックのほかにも、JavaScriptを使っていろいろなチェックができます。次の方法で、郵便番号やメールアドレスのチェックができます。

郵便番号

入力した値が、郵便番号の形式になっているかをチェックしてみます。それには、次のように形式パターンを管理するRegExpオブジェクトを作成します。

Sample4-2-2.html ■ 郵便番号のパターンをチェックする

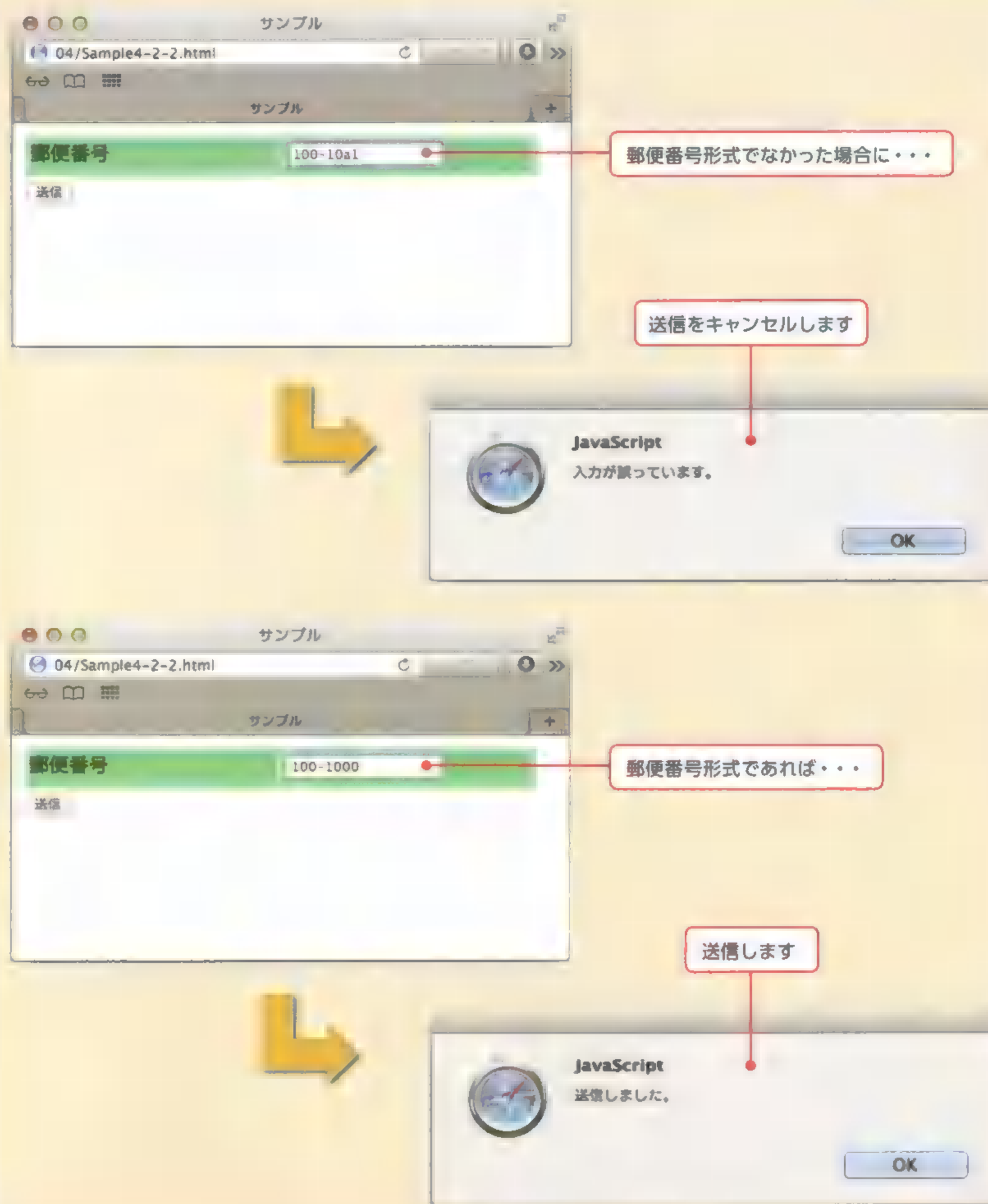
```
<!DOCTYPE html>
<html lang="ja">
<head>
<link rel="stylesheet" href="Form.css">
<title>サンプル</title>
<script type="text/javascript">
```

```
function check()
{
   var p = new RegExp("^[0-9]{3}-[0-9]{4}$", "i");
   var s = myform.address.value;
   var res = s.match(p);

  if(res == null){
    window.alert("入力が誤っています。");
     res =  false;
   }
   else{
      window.alert("送信しました。");
      res = true;
   }
   return res;
}
</script>
</head>
<form name="myform" action="Sample4-2-2.html" onsubmit="return check()">
<table>
<tr>
<td>郵便番号</td>
<td><input type="text" name ="address"/></td>
</tr>
</table>
<input type="submit" value="送信"/>
</form>
</body>
</html>
```

郵便番号形式であるかをチェックして送信することができます

Stringオブジェクトのmatch()メソッドは、入力データがRegExpオブジェクトのパターンが一致しなかった場合に、nullという結果を戻します。そこで、ここでは結果がnullであるかどうかを調べて、データのチェックをしているのです。

「null(ヌル)」は、オブジェクトが存在しないことをあらわす指定です。プログラムの世界ではよく使われる指定となっています。

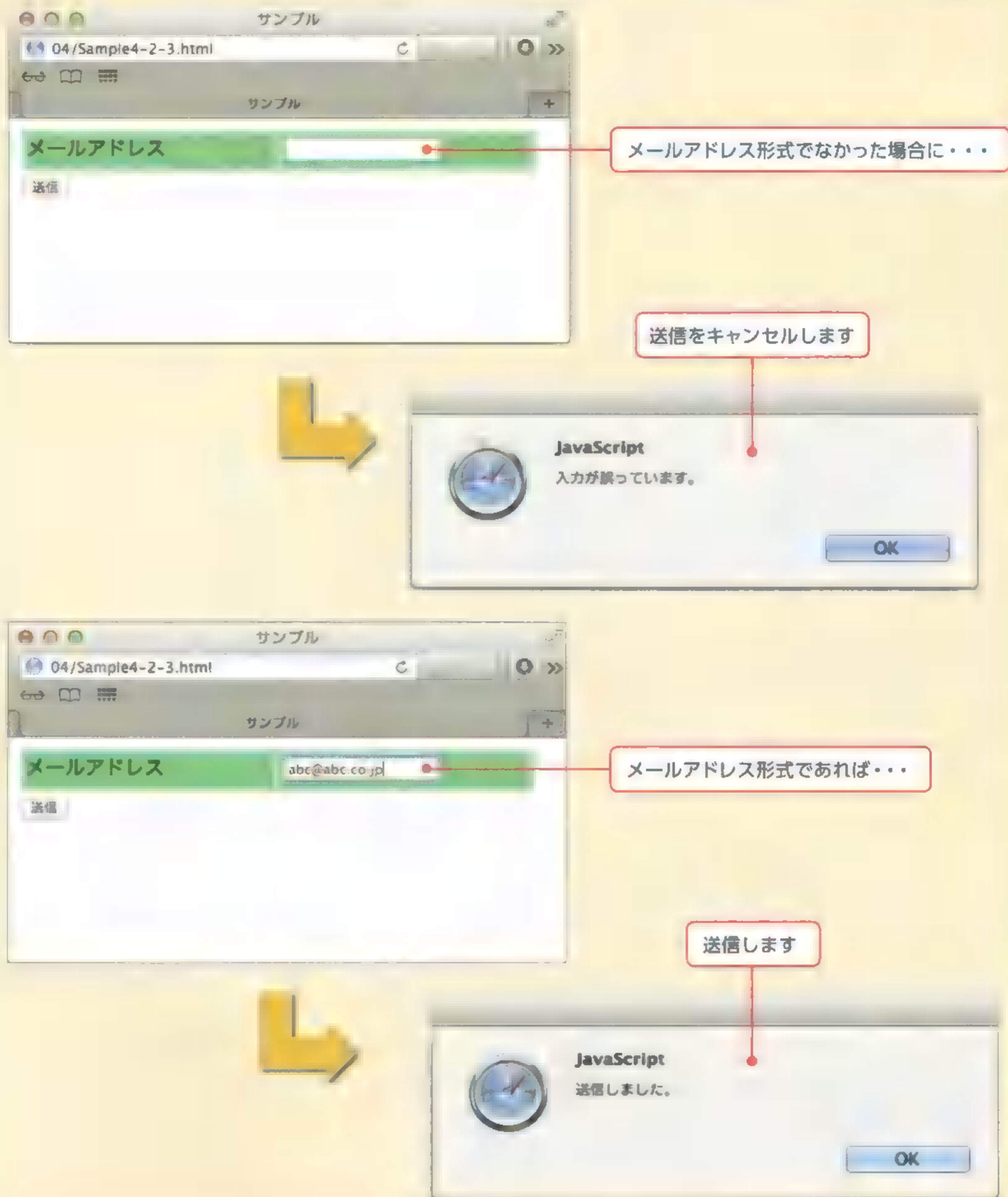
■ Sample4-2-2の表示

メールアドレス

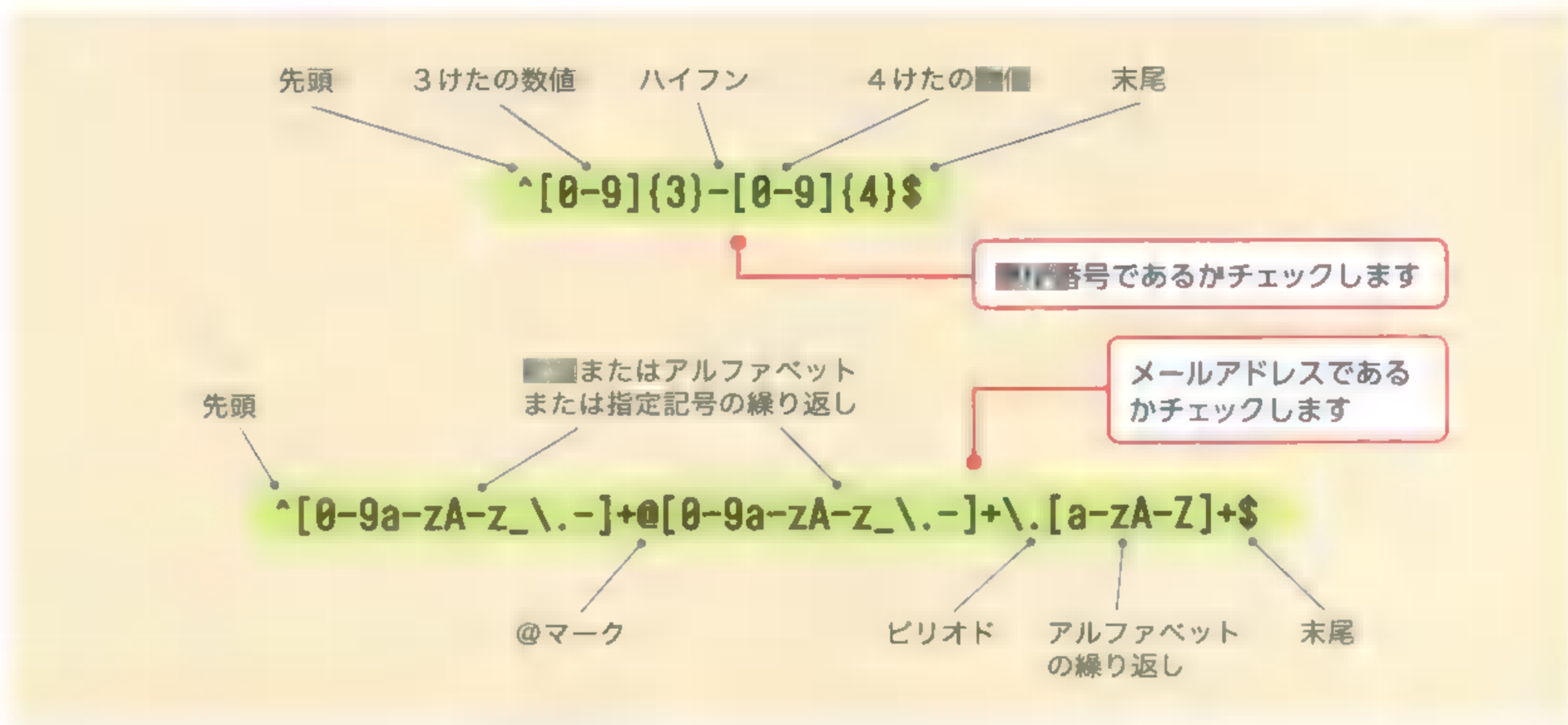
メールアドレスも同様の方法でチェックすることができます。メールアドレスの形式になっているかどうかをチェックするには、check()処理内を次のようにします。

Sample4-2-3.html ■ メールアドレスのパターンをチェックする

```
<!DOCTYPE html>
<html lang="ja">
<head>
<link rel="stylesheet" href="Form.css">
<title>サンプル</title>
<script type="text/javascript">
function check()
{
   var p = new RegExp("^[0-9a-zA-z_\.-]+@[0-9a-zA-z_\.-]+\.[a-zA-Z]+$", "i");
   var s = myform.address.value;
   var res = s.match(p);

  if(res == null){
    window.alert("入力が誤っています。");
     res =  false;
   }
   else{
      window.alert("送信しました。");
      res = true;
   }
   return res;
}
</script>
</head>
<body>
<body>
<form name="myform" action="Sample4-2-3.html" onsubmit="return check()">
<table>
<tr>
<td>メールアドレス</td>
<td><input type="text" name ="address"/></td>
</tr>
</table>
<input type="submit" value="送信"/>
</form>
</body>
</html>
```

メールアドレス形式であるかをチェックして送信することができます

■ Sample4-2-3の表示

パターンを指定してチェックする

ここでは次のようにパターンを指定して、メールアドレスや郵便番号のチェックを行っています。

パターンを作成するためには、高度な知識が必要ですが、代表的なパターンをいくつか覚えて使えるようになっておくと便利です。たとえば、次のようなパターンを使ってフォームに入力されたデータの形式を調べることができます。

■ パターンの例

パターン	パターンの意味	あてはまる文字列の例
[012345]	012345のいずれか	3
[0-9]	0～9のいずれか	5
[A-Z]	A～Zのいずれか	B
[A-Za-z]	A～Z、a～zのいずれか	b
[^012345]	012345ではない文字	6
[01][01]	00、01、10、11のいずれか	01
[A-Za-z][0-9]	アルファベット1つに数字が1つ続く	A0

正規表現

パターンをつくる表記は、正規表現と呼ばれています。ここで使ったRegExpオブジェクトは、正規表現の機能をまとめたオブジェクトです。JavaScriptの正規表現のくわしい利用方法については、『やさしいWebアプリプログラミング』(SBクリエイティブ) などを参考にしてみてください。

4-3 選択商品を表示しよう

チャレンジ!! 選んだ商品を表示しよう

今度は、フォームを使ってユーザーに商品を選択させ、選択した商品をWebページ上に確認表示するようにしてみましょう。フォーム部品のリストボックスメニューを使って商品を選び、表に表示することにします。

■ 確認表示する

商品を表に追加するには？

選択フォームを設計する

これまでの知識を生かして作成してみましょう。まず、入力データを受け付けるためのフォームを設計します。

フォーム上のボタンをクリックしたときに、add()にまとめた処理を行うようにしましょう。

フォームの下には表を付け加えることができるように、「table」というID名をつけた表を配置しておきます。

```
<form name="myform" action="Sample4-3-1.html">
<select name ="items" size="3">
<option value="りんご">りんご</option>
<option value="みかん">みかん</option>
<option value="もも">もも</option>
```

```
</select>
<input type="button" value="追加" onclick="add()"/>
</form>
<table id="table">
</table>
```

クリック時に処理します

この表に表示します

このように名前をつけておけば、フォームの値は次のように得ることができます。

```
var itm = myform.items.value;
```

フォームのデータを得ることができます

■ フォームの値を得る

表に1行追加する

表に1行付け加えるため、表のIDを指定してDOMオブジェクトを取得します（❶）。追加する「tr」要素（行）、「td」要素（セル）を作成しましょう（❷・❸）。さらに、フォームの値を指定してテキストを作成します（❹）。

最後に、作成したテキストをセルに追加し、これをさらに行、テーブルの順に追加することにします（❺）。

```
var e = document.getElementById("table");
var elmr = document.createElement("tr");
var elmd = document.createElement("td");
var txt = document.createTextNode(itm);
elmd.appendChild(txt);
elmr.appendChild(elmd);
e.appendChild(elmr);
```

■ 表に1行追加する

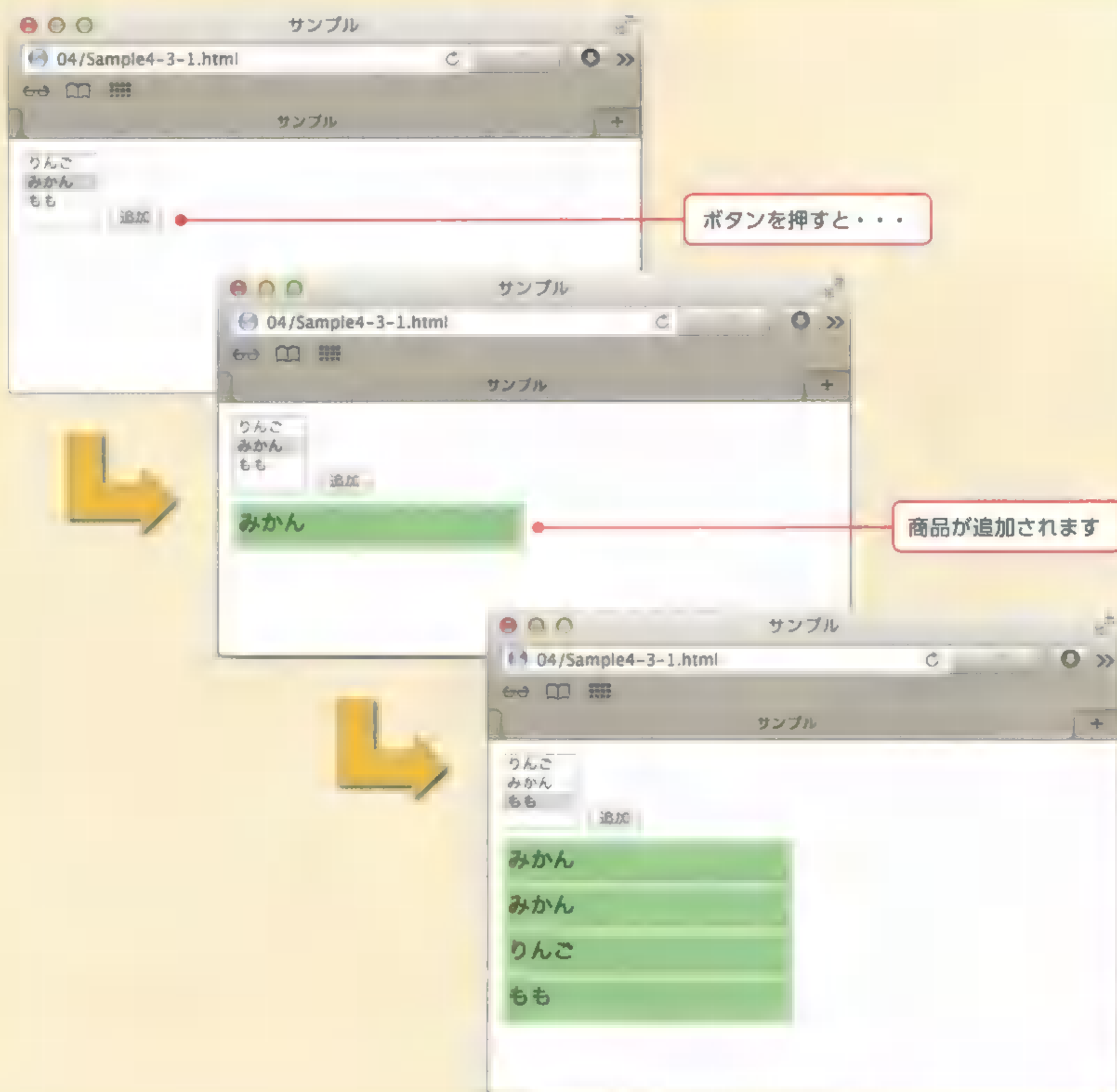
選択商品を表示しよう

それでは、手順をまとめてページを作成してみましょう。追加ボタンを押すたびに、商品名が表に追加されることがわかるでしょう。

Sample4-3-1.html ■ 商品名を表示する

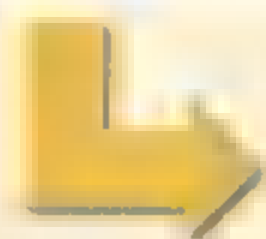
```
<!DOCTYPE html>
<html lang="ja">
<head>
<link rel="stylesheet" href="Form.css">
<title>サンプル</title>
<script type="text/javascript">
var itemlist = ["りんご", "みかん", "もも"];
function add()
{
   var itm = myform.items.value;

   var e = document.getElementById("table");
   var elmr = document.createElement("tr");      ← 1行追加します
   var elmd = document.createElement("td");
   var txt = document.createTextNode(itm);
   elmd.appendChild(txt);
   elmr.appendChild(elmd);
   e.appendChild(elmr);
}
</script>
</head>
<body>
<form name="myform" action="Sample4-3-1.html">
<select name ="items" size="3">
<option value="りんご">りんご</option>
```

```
<option value="みかん">みかん</option>
<option value="もも">もも</option>
</select>
<input type="button" value="追加" onclick="add()"/>
</form>
<table id="table">
</table>
</body>
</html>
```

■ Sample4-3-1 の表示

4-4 合計金額を計算しよう

チャレンジ!! 商品価格を計算しよう

商品を確認表示することができたので、今度は選択した商品と個数について価格を調べ、合計額を計算して確認表示する機能を作成してみましょう。

■ 計算と確認

商品名	価格	個数	計
りんご	500	1	500
みかん	300	1	300
もも	200	3	600

価格表を用意する

まず、商品と価格の対応をあらわす価格表を考えてみましょう。ここでは、価格表をあらわすために、JavaScriptの連想配列というものを使うことにします。連想配列は、名前を指定して値を取り出すことができる配列です。「名前：値」というデータを格納します。この配列を使うと、名前を指定して値を取り出すことができます。ここでは、商品名を指定して価格を取り出すことができるわけです。たとえば、「りんご」を指定すれば「500」を取り出すことができるでしょう。

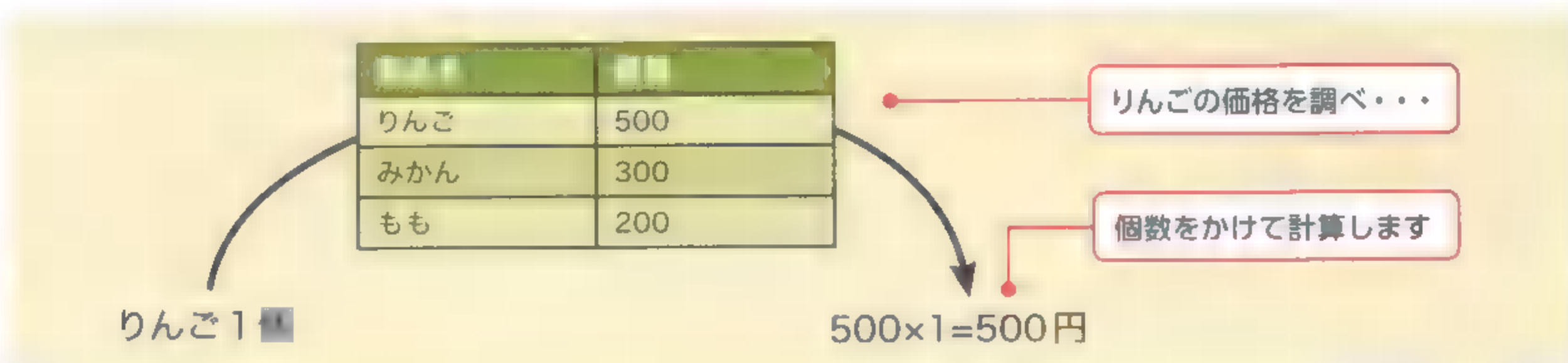
■ 価格表

商品名	価格
りんご	500
みかん	300
もも	200

```
var pricelist
  = {"りんご":500, "みかん":300, "もも":200};
```

「商品名：価格」とします

小計を計算する

商品名が選択されたら、連想配列から該当する商品の価格を調べ、選択されている個数とかけあわせて金額を計算することにします。たとえば、「りんご」「1」が選択された場合には、りんごの価格500円を調べて、500 × 1 = 500円と計算するのです。

小計を計算する

この計算作業をプログラムにしてみましょう。連想配列priceList[]を価格表、変数itmを商品名、変数nを個数とすると、次のように考えることができます。

```
var sum = 0;

function add()
{

   var c = pricelist[itm]*n;
   sum = sum + c;

}
```

また、選択追加するたびに、合計についても計算しておくことにします。計算した合計金額は、合計ボタンを押したときに表示するようにしておきましょう。

合計を記憶するための変数sumは、処理の外側で準備します。これは処理が終わっても、現在の合計額を保持しておくためです。処理が終わっても変数に値を記録しておきたい場合には、処理の外側で変数を準備します。

選択をクリアする

クリア機能をつけておきましょう。「table」ノードの下にある子ノードを調べ、最後の子ノードから順番に2つ目の子ノードまでをDOMによって削除します。これで表題以外の行が削除されるようになります。

```
var e = document.getElementById("table");
for(var i=e.childNodes.length-1; i>1; i--){
  e.removeChild(e.childNodes[i]);
}
```


計算プログラムを完成させる

最後にプログラムを完成させてみましょう。

Sample4-4-1.html ■ 価格を計算する

```
<!DOCTYPE html>
<html lang="ja">
<head>
<link rel="stylesheet" href="Form.css">
<title>サンプル</title>
<script type="text/javascript">
var sum = 0;
var itemlist = ["りんご", "みかん", "もも"];
var pricelist = {"りんご":500, "みかん":300, "もも":200};
function add()
{
   var itm = myform.items.value;
   var n = myform.num.value;

   var e = document.getElementById("table");

   var elmr = document.createElement("tr");

   var elmd = document.createElement("td");
   var txt = document.createTextNode(itm);
   elmd.appendChild(txt);
   elmr.appendChild(elmd);

   var elmd = document.createElement("td");
   var txt = document.createTextNode(pricelist[itm]);
```

4

JavaScriptでチェックしよう!

```
    elmd.appendChild(txt);
    elmr.appendChild(elmd);

    var elmd = document.createElement("td");
    var txt = document.createTextNode(n);
    elmd.appendChild(txt);
    elmr.appendChild(elmd);

    var c = pricelist[itm]*n;
    sum = sum + c;
    var elmd = document.createElement("td");
    var txt = document.createTextNode(c);
    elmd.appendChild(txt);
    elmr.appendChild(elmd);

    e.appendChild(elmr);
}
function cls()
{
    var e = document.getElementById("table");
    for(var i=e.childNodes.length-1; i>1; i--){
      e.removeChild(e.childNodes[i]);
    }
    sum = 0;
}
function calc()
{
    window.alert("合計:" + sum + "円");
}
</script>
</head>
<body>
<form name="myform" action="Sample4-4-1.html">
<select name ="items">
<option value="りんご">りんご</option>
<option value="みかん">みかん</option>
<option value="もも">もも</option>
</select>
<select name ="num">
<option value=1>1</option>
<option value=2>2</option>
<option value=3>3</option>
</select>
<input type="button" value="追加" onclick="add()"/>
<input type="button" value="クリア" onclick="cls()"/>
<input type="button" value="合計" onclick="calc()"/>
</form>
```

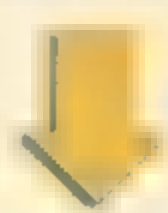
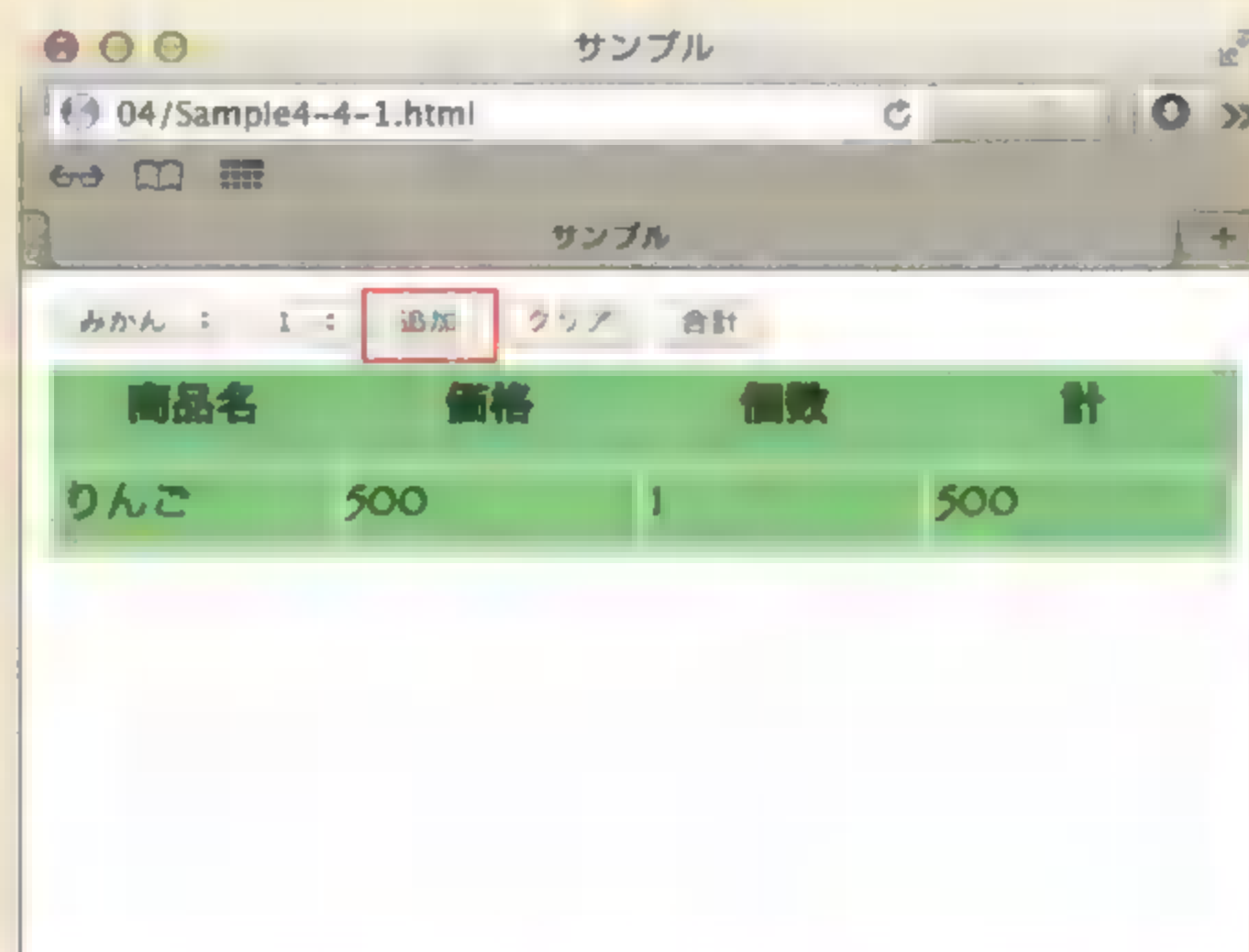
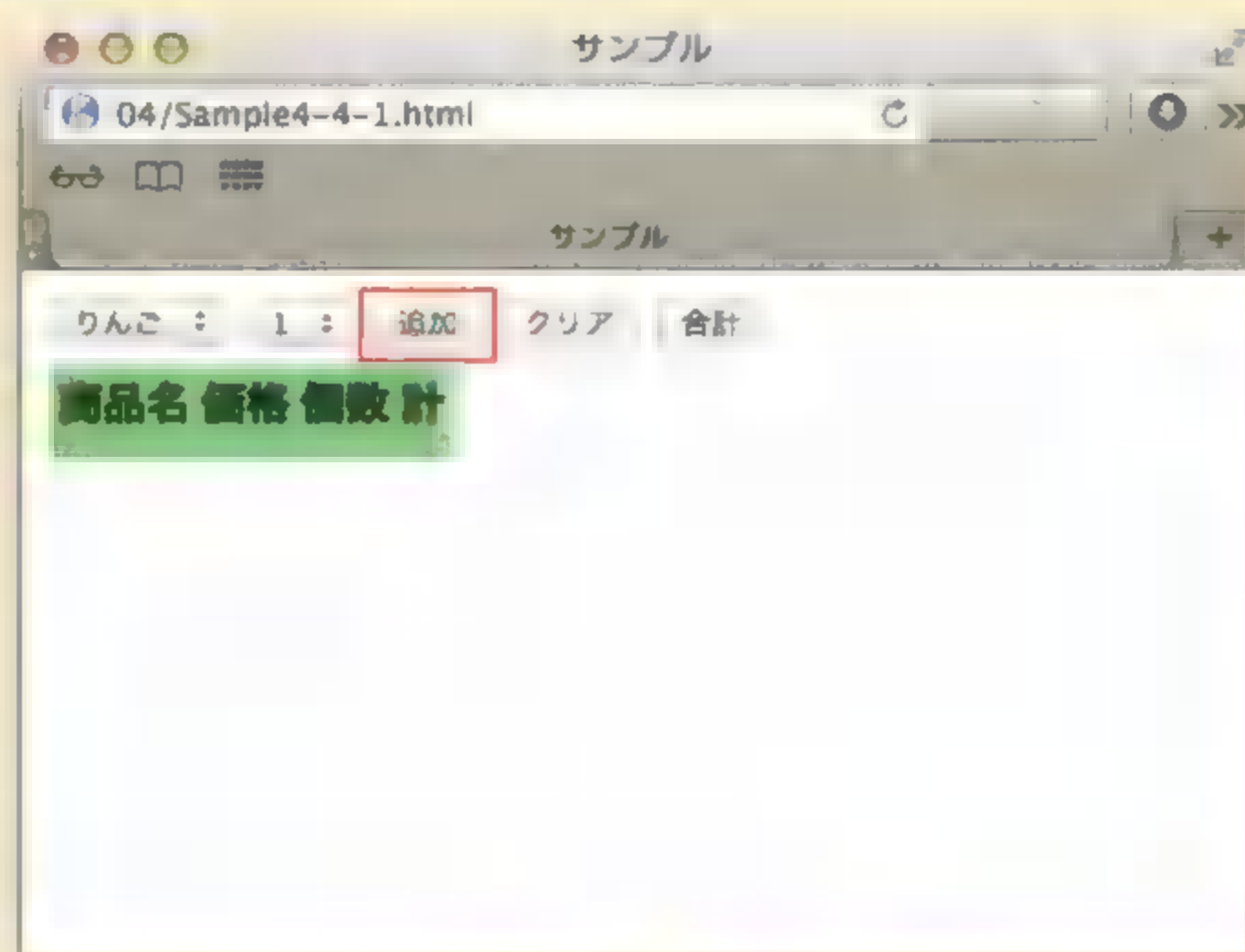
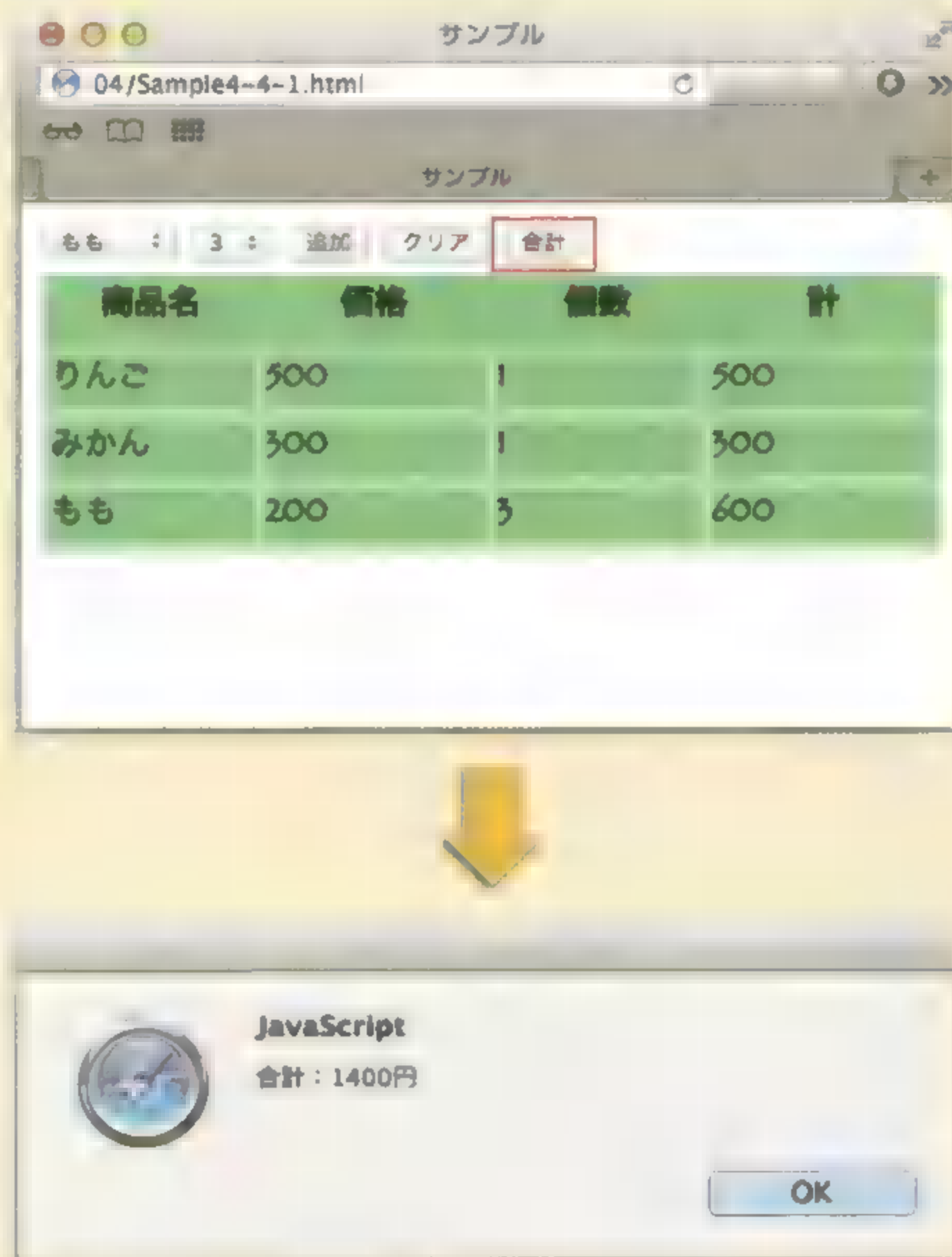
小計を表示します

合計額を計算します

表をクリアします

合計額を表示します

```
<table id="table">
<tr>
<th>商品名</th><th>価格</th><th>個数</th><th>計</th>
</tr>
</table>
</body>
</html>
```

■ Sample4-4-1 の表示

小計や合計が計算できることを確認してみてください。

データベース

ここでは、ごく簡単な商品のデータを扱いました。実際に商品データや価格データを調べたり、ユーザーからの注文データを受け取るには、もっと大量のデータを扱うことになるでしょう。

このような大量のデータを処理する際には、ファイルやデータベースを使います。

データを扱うファイルとしてよく使われる形式として、テキストファイル・XMLファイル・JSONファイルなどがあります。また、特に大量のデータを扱う際には、Webサーバー側でデータベースを使うことが多くなっています。Webサーバー側のデータベースを扱うためには、サーバー上で動作するプログラムを作成することが必要です。

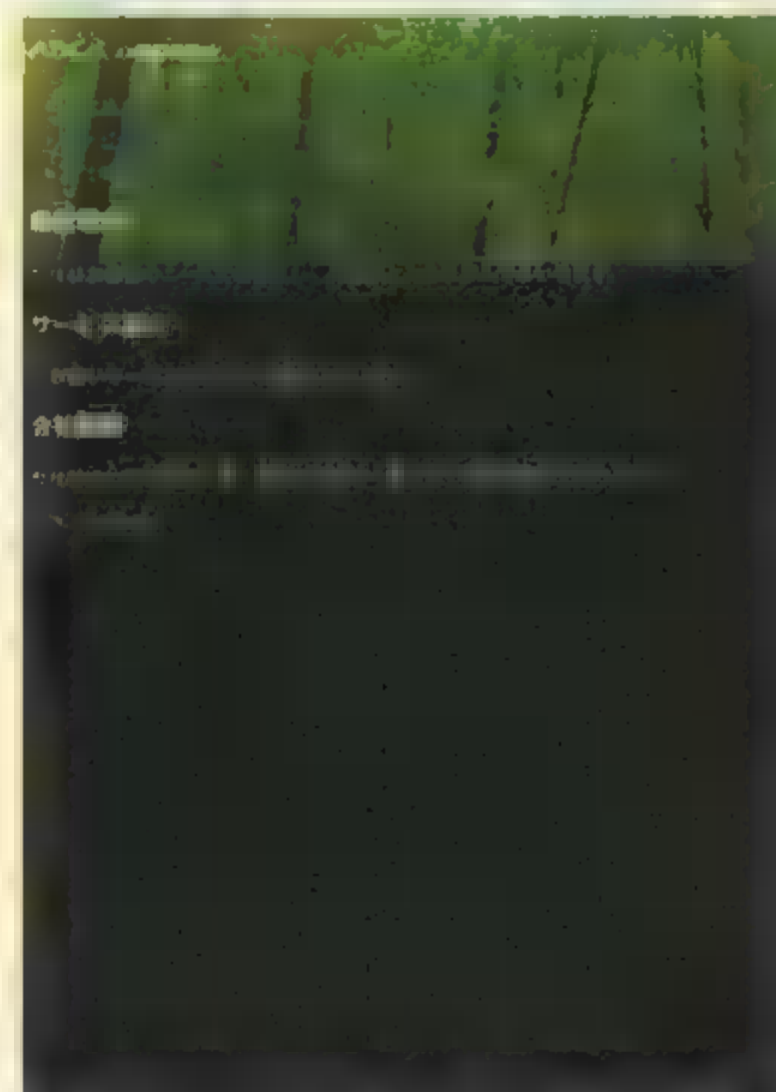
JavaScriptでは、入力のチェックや計算を行ったり、Webページに表示する処理を行うことが多くなっています。

4-5 端末に応じたサイトにしよう

チャレンジ! 端末別の表示にしてみよう

この章の最後に、JavaScriptでユーザーの使用するブラウザに適したサイトを構築してみることにしましょう。ユーザーの使用しているブラウザ環境をチェックして処理をするのです。

会社や自宅のPCばかりでなく、外出先で利用するスマートフォンなどさまざまな場所でWebサイトにアクセスすることができます。JavaScriptを使えば、こうした環境にきめこまかに対応するWebサイトを構築することができます。

■ PCとスマートフォン

Step 1 Webページの内容を検討する

まず、基本となるWebページの内容を検討していくことにします。ここでは、一例として、次の要素をもつページを作成してみましょう。ヘッダ・ナビゲーション・内容・フッタを構築することにします。

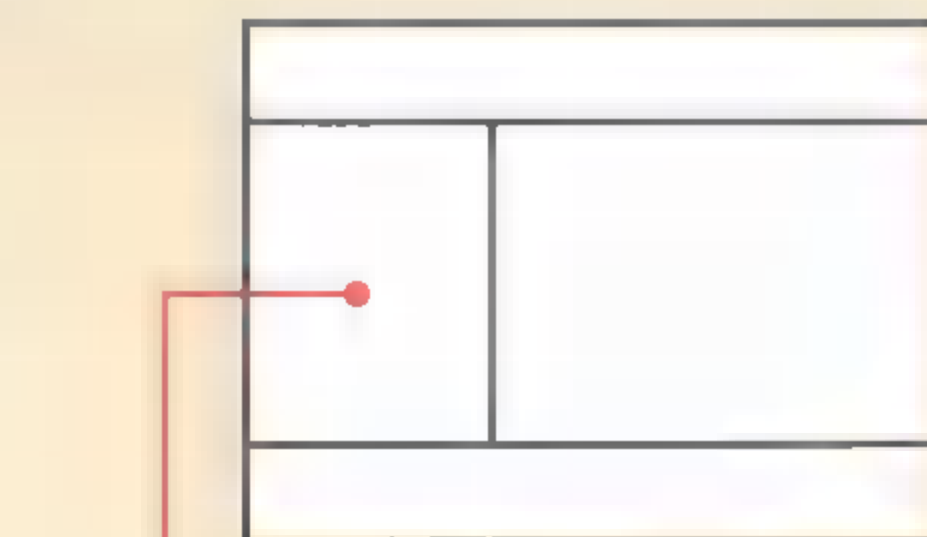
```
<header>
<body>
<h1>ABC Company</h1>

<nav>
<ul>
<li><a href="#goal">経営理念</a>
<li><a href="#service">サービス紹介</a>
<li><a href="#company">会社概要</a>
</ul>
</nav>
</header>

<div id="content">
<div id="goal">
<h2>経営理念</h2>
<p>弊社はお[illegible]様のニーズにお答えし、お[illegible]とともに歩んでまいります。</p>
</div>

<div id="service">
<h2>サービス紹介</h2>
<p>お取扱いサービスについてご紹介いたします。</p>
</div>

<div id="company">
<h2>会社概要</h2>
<p>弊社は1980年の設立以来、地域の皆様の一員として営業を続けております。</p>
</div>

</div>

<footer>
<address>
ABC Company
</address>
</footer>

</body>
</html>
```

■ 基本となるWebページの内容

Step2 レイアウトを設計する

それでは、検討した内容のページを、PCとスマートフォンのそれぞれの特性を生かしたWebページのレイアウトで表示するように考えていきましょう。

PCでは大きな画面を生かしたレイアウトに、スマートフォンでは小さな画面をうまく利用するレイアウトを検討してみましょう。ここでは、次のようなレイアウトにすることを検討してみます。

■ PC（左）とスマートフォン（右）のレイアウトを考える

PCのレイアウトでは、サイトのナビゲーションを左に、内容を右において2段組みとすることにします。スマートフォンではリスト形式にしましょう。

Step 3　2種類のレイアウトを作成する

それでは、このレイアウトに対応するCSSを作成します。スマートフォン用のページとPC用の2つのレイアウトを、それぞれCSSファイルとして実際に用意してみましょう。

SamplePC.css

```
body{
    background-color: #000000;
    color: #FFFFFF;
    font-family: fantasy;
    background-image: url(title.jpg);
    background-repeat: repeat-x;
}
h1{
    font-size: 2em;
    font-family: fantasy;
}
nav{
    color: #FFFFF0;
    float: left;
}

#content{
    float: right;
}

footer{
    clear: both;
}
```

PC用のレイアウトを作成します

SamplePhone.css

```
body{
    background-color: #000000;
    color: #FFFFFF;
    font-family: fantasy;
    background-image: url(title.jpg);
    background-repeat: repeat-x;
}
h1{
    font-size: 2em;
    font-family: fantasy;
}
nav{
```

スマートフォン用のレイアウトを作成します

```
    color: #FFFFF0;
```

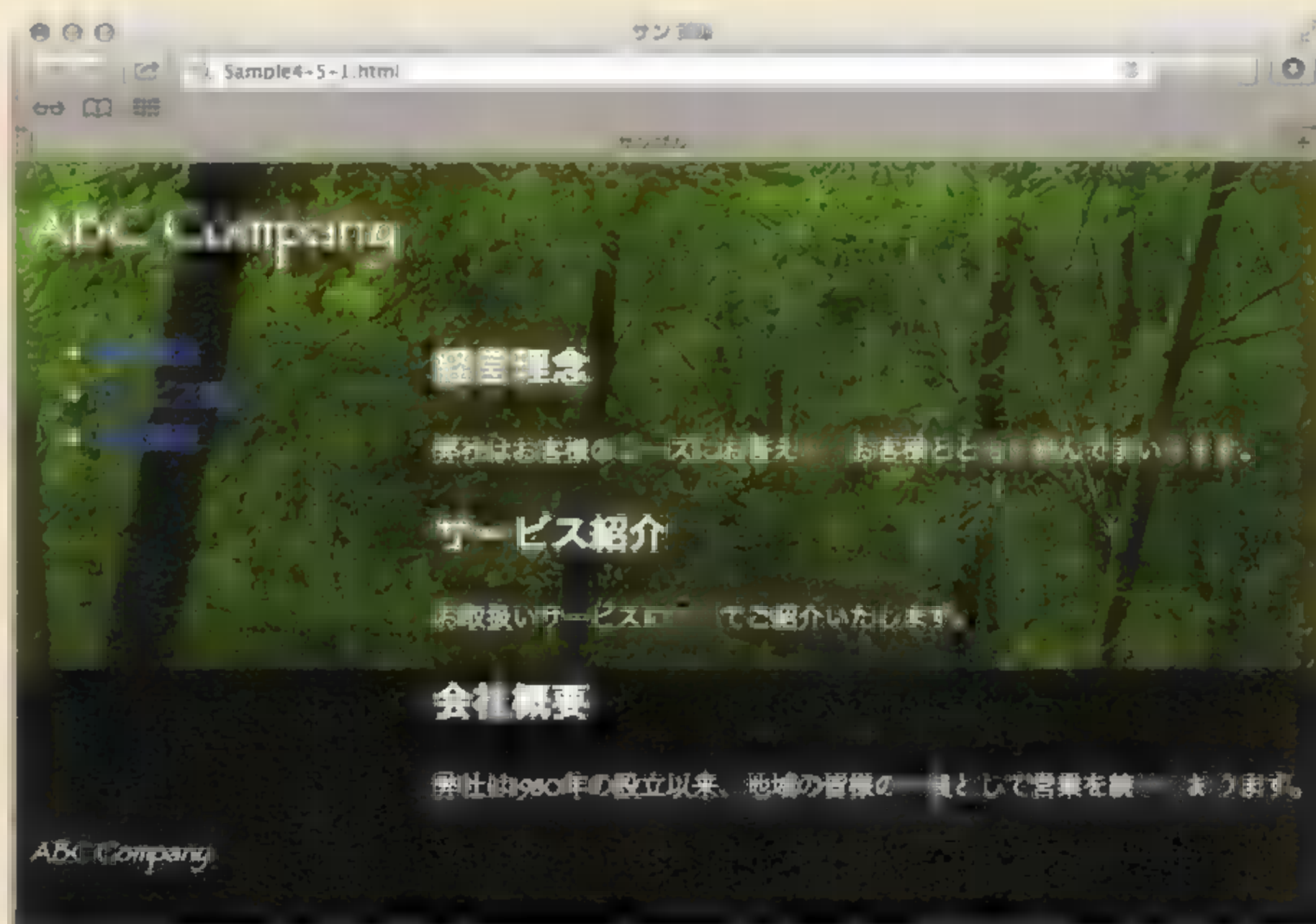
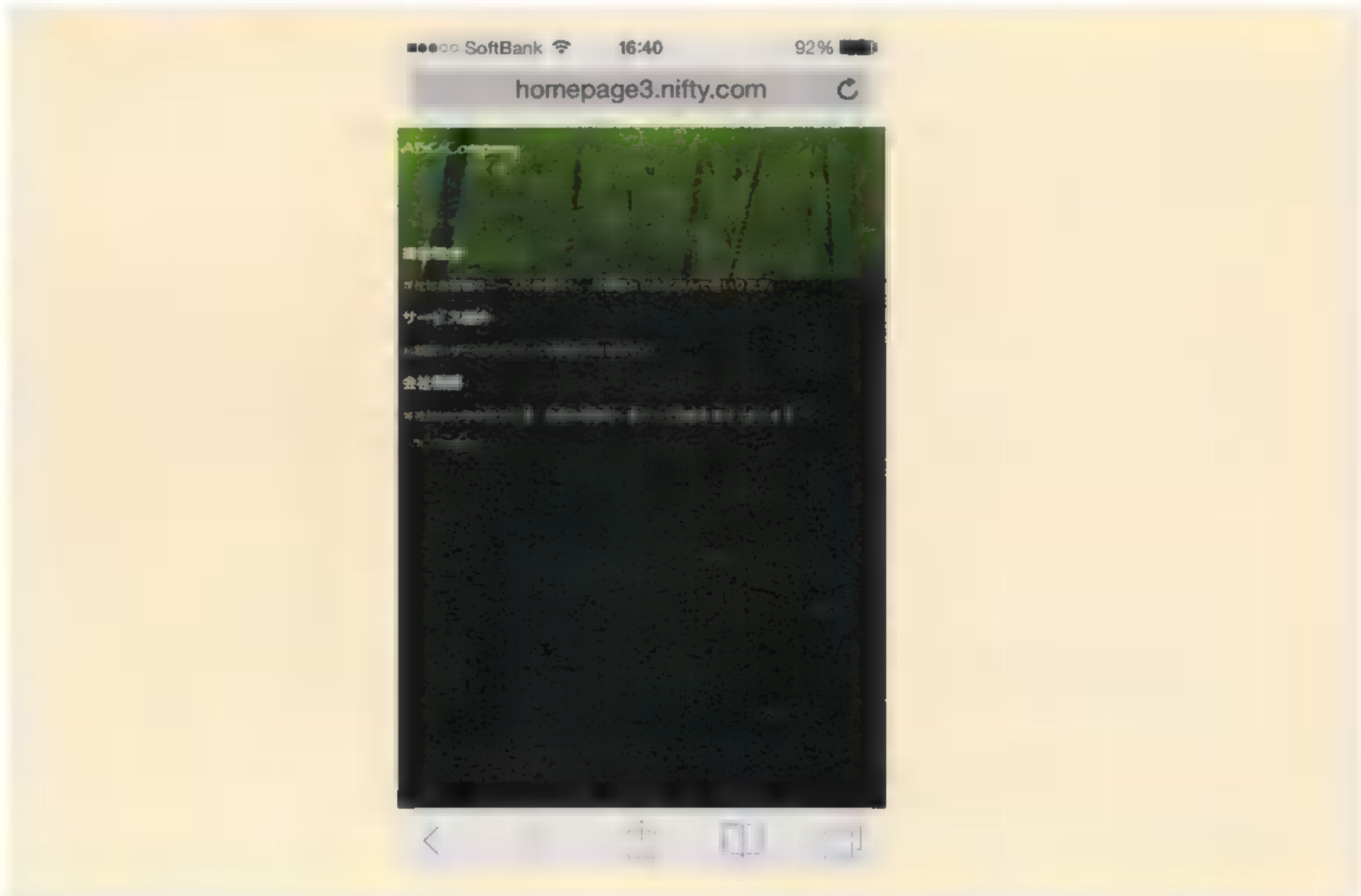
Step 4 端末別のレイアウトで表示する

それでは、実際のページを作成してみましょう。

Sample4-5-1.html ■ 端末別のページ

```
<!DOCTYPE html>
<html lang="ja">
<head>
<title>サンプル</title>
<script type="text/javascript">
   var ■ = navigator.userAgent;
   var ios = a.indexOf("iPhone")>0 || a.indexOf("iPad")>0 || a.indexOf("iPod")>0;
   var android = a.indexOf("Android")>0;

if(ios == true || android == true){
   document.writeln('<link rel="stylesheet" href="SamplePhone.css">');
}
else{
   document.writeln('<link rel="stylesheet" href="SamplePC.css">');
}
</script>
</head>

<header>
<body>
<h1>ABC Company</h1>
<nav>
<ul>
<li><a href="#goal">経営理念</a>
<li><a href="#service">サービス紹介</a>
<li><a href="#company">会社概要</a>
</ul>
</nav>
</header>

<div id="content">
<div id="goal">
<h2>経営理念</h2>
<p>弊社はお客様のニーズにお答えし、お客様とともに歩んでまいります。</p>
</div>
```

```
<div id="service">
<h2>サービス紹介</h2>
<p>お取扱いサービスについてご紹介いたします。</p>
</div>

<div id="company">
<h2>会社概要</h2>
<p>弊社は1980年の設立以来、地域の皆様の一員として営業を続けております。</p>
</div>

</div>

<footer>
<address>
ABC Company
</address>
</footer>

</body>
</html>
```

■ Sample4-5-1 の表示（PC）

■ Sample4-5-1の表示（スマートフォン）

ブラウザのチェックをする

ユーザーが使用しているブラウザをチェックするには、NavigatorオブジェクトのuserAgentプロパティを確認します。このuserAgentプロパティであらわされるデータによって、使用中のブラウザの種類を調べることができます。ここではいったん変数に記憶し、変数の値にiPhone、iPad、iPod、Androidの文字列が入っているかどうかをチェックし、スタイルシートの名前を指定しているのです。

```
var ■ = navigator.userAgent;
var ios = a.indexOf("iPhone")>0 || a.indexOf("iPad")>0 || a.indexOf("iPod")>0;
var android =  a.indexOf("Android")>0;
```

Tips

indexOf()は、指定した文字列の位置を調べる機能です。0以上である場合には、その文字列が入っていることになります。つまり、iPhone、iPad、iPod、Androidの文字列が入っているかを調べます。入っていればスマートフォンであることになります。

Navigatorオブジェクト

Navigatorオブジェクトは、ブラウザに関する情報を取得することができます。コード中では、「navigator」の表記でオブジェクトを扱うことができるようになっています。

■ Navigatorオブジェクトの主なメンバ

メンバ	説明
appName	ブラウザ名
appversion	ブラウザバージョン
platform	プラットフォーム
userAgent	ユーザーエージェント（ブラウザ）

Step 5 高機能なプログラムを作成するためには

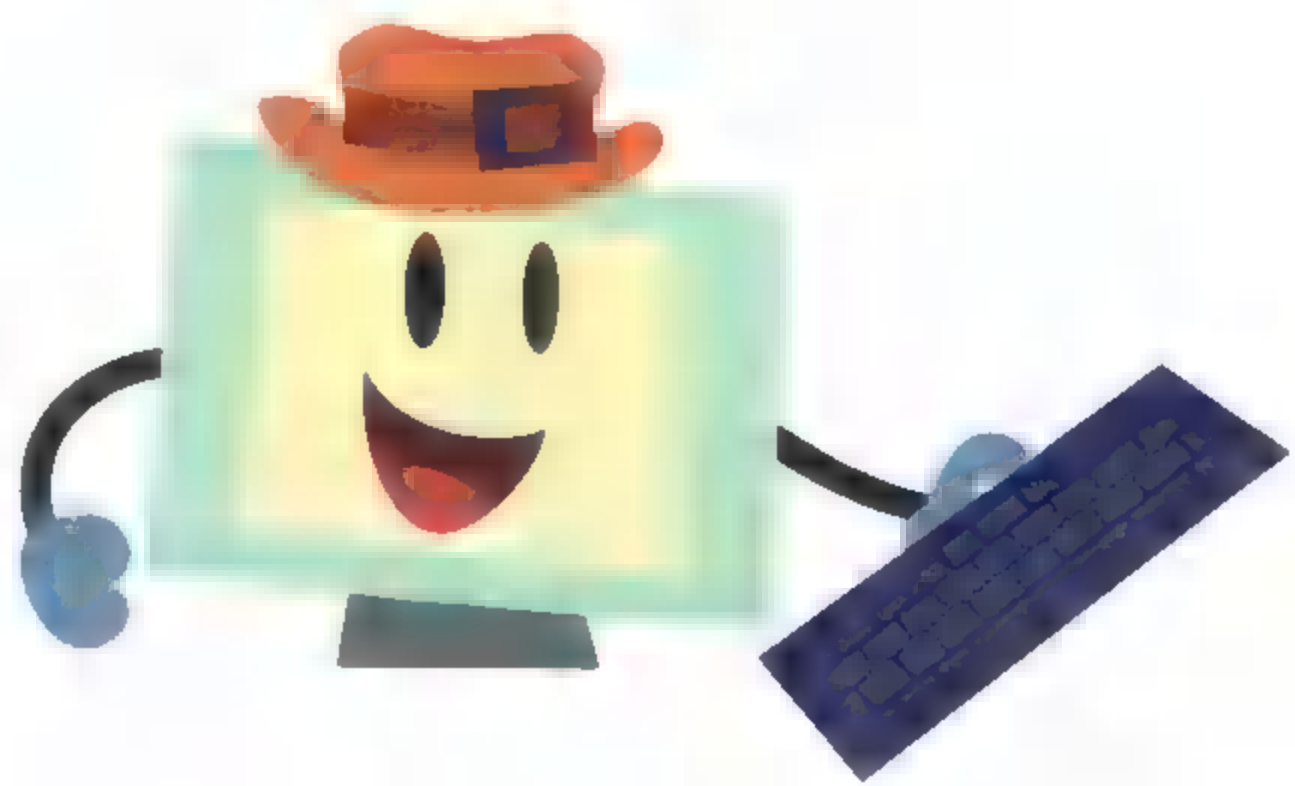
これまで、JavaScriptのさまざまな機能を学び、活用してきました。JavaScriptには、特に便利な機能をまとめたオブジェクトがまとめられ、公開されています。こうして提供されるオブジェクトの集まりは、ライブラリ（library）と呼ばれています。

さらに高度なプログラムを作成していくためには、ライブラリを利用することもできるでしょう。さまざまな機能を活用して実践的なプログラムを検討してみてください。

■ ライブラリ

jQuery

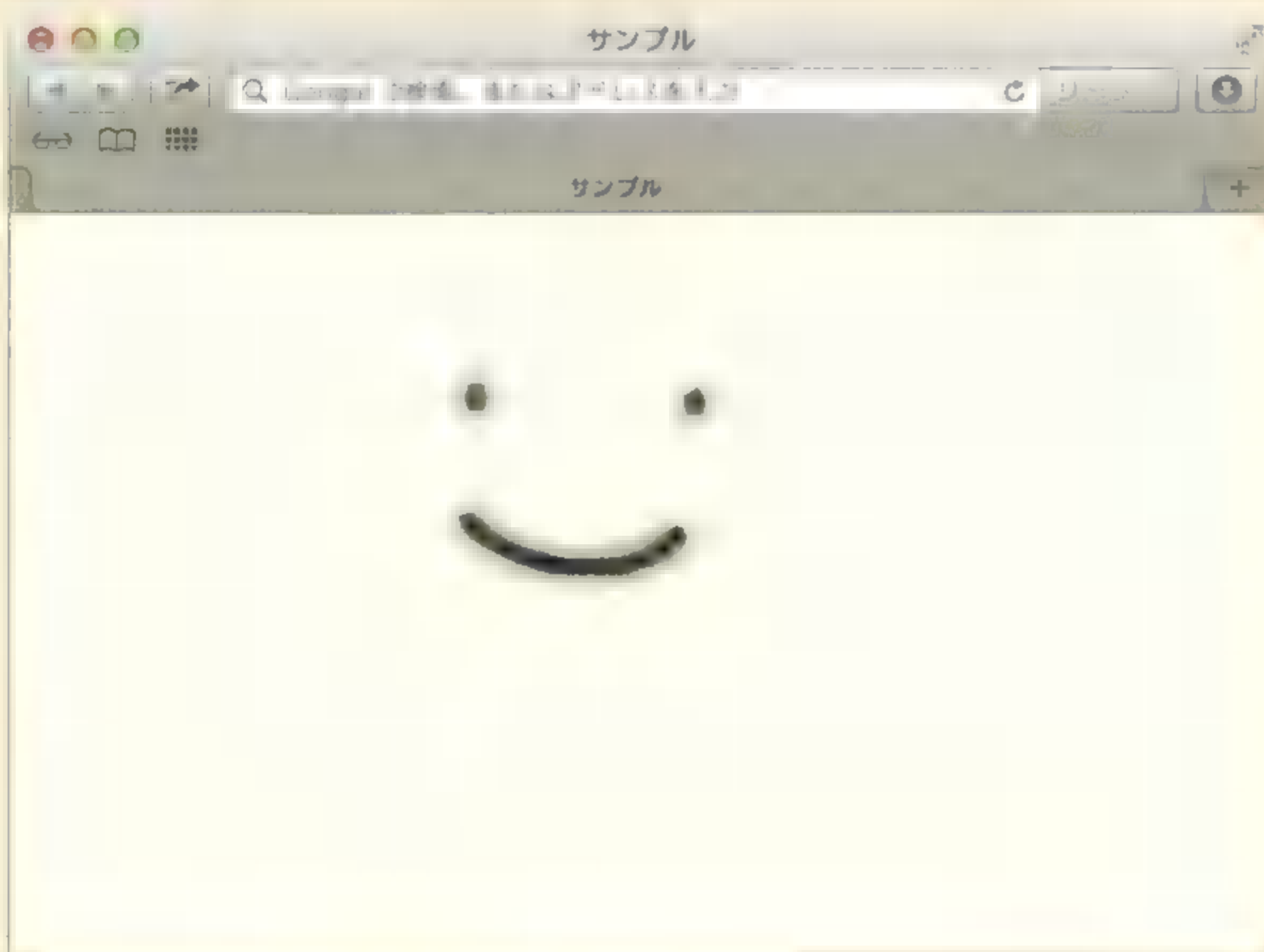
Webサイトを構築するための一般的なライブラリとしては、jQueryがよく利用されています。よりバリエーションに富んだプログラムを作成していくためには、こうしたライブラリを活用することも検討してみるとよいでしょう。なお、本書付録ではjQueryのかんたんな使い方について紹介しています。

■ jQueryのWebページ（http://jquery.com/）

4 JavaScriptでチェックしよう!

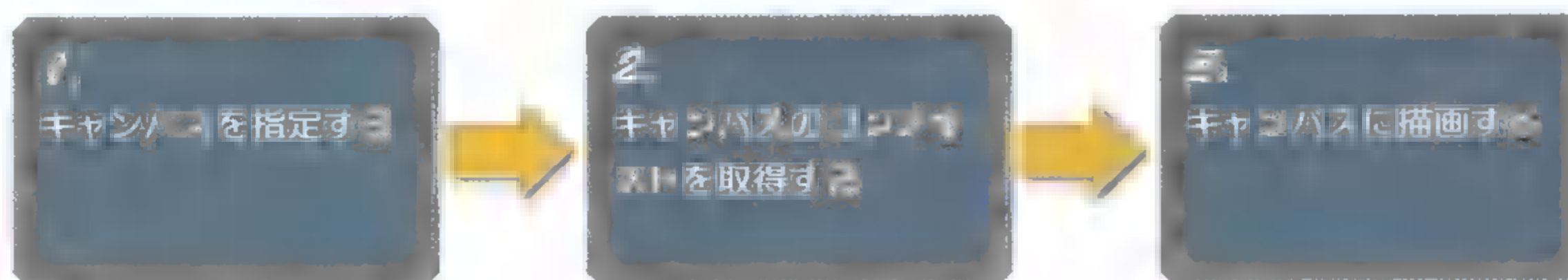
5-1 キャンバスを使おう！

チャレンジ！ キャンバスに描画してみよう

JavaScriptを利用すれば、Webページ上にグラフィックを描くプログラムを作成することができます。描画をするためのエリアは**キャンバス**（Canvas）と呼ばれています。ここではキャンバスを使って描画するページを作成してみましょう。

■ キャンバス

グラフィックを描画する機能は、主にHTML5の機能の一部として提供されています。JavaScriptでは、このキャンバスをDOMによって操作しています。

キャンバスを使うには？

それでは、キャンバスを使ってみましょう。キャンバスは次のように使用します。

キャンバスを指定する

キャンバスはHTML5で追加された機能です。キャンバスをJavaScriptで扱うには、DOMによってキャンバス要素を指定し、操作します。

次のようにキャンバスをあらわすDOMオブジェクトを指定しましょう。HTML文書中の「canvas」要素に、あらかじめID名をつけておきます。ここでは、canvasというID名をつけることにしました。

```
var c = document.getElementById("canvas");
...
<canvas id="canvas"></canvas>
```

コンテキストを取得する

キャンバスを指定したら、描画を行うために必要になるコンテキストと呼ばれるオブジェクトを取得します。平面図形を描画するコンテキストオブジェクトを得るためには、キャンバスのgetContext()メソッドを使って、2dという文字列を指定してください。

コンテキストを取得します

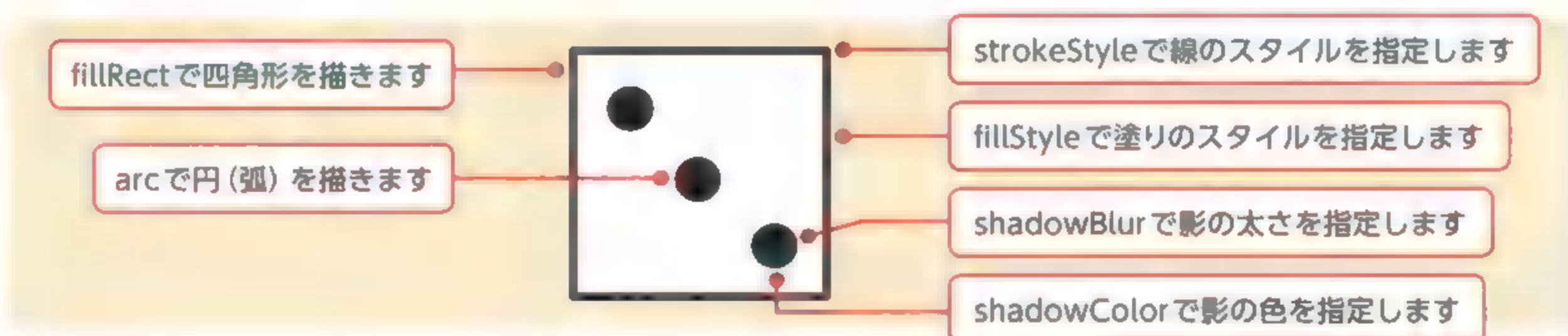
```
var cnt = c.getContext("2d");
```

キャンバスに描画する

コンテキストを取得したら描画を行います。描画する影の太さ・色の設定、線の色の設定などを行ってから描画します。

■描画に使う主なメンバ

メンバ	内容
shadowBlur	影の太さ
shadowColor	影の色
strokeStyle	線のスタイル
fillStyle	塗りのスタイル
fillRect()	塗りつぶし四角形
arc()	円弧
beginPath()	パスを開始する
closePath()	パスを終了する
fill()	塗りつぶす

たとえば、サイコロを描く場合には、次の指定を使うことができるでしょう。線の太さ・スタイルを指定し、四角形・円を描きます。影の太さと色も指定できます。

サイコロを描画する

実際に、四角形と円を組み合わせた図形（サイコロの図）を描画してみましょう。この章では、次のスタイルシートを使います。

Canvas.css ■ キャンバス用のスタイルシート

```
body{
   background-color: #000000;
}
```

Sample5-1-1.html ■ サイコロを描画する

```
<!DOCTYPE html>
<html lang="ja">
<head>
<link rel="stylesheet" href="Canvas.css">
<title>サンプル</title>
<script type="text/javascript">
```

```
function init()
{
    var c = document.getElementById("canvas");
    var cnt = c.getContext("2d");

    cnt.shadowBlur = 10;
    cnt.shadowColor = "#000000";

    cnt.strokeStyle = "#000000";
    cnt.fillStyle = "#FFFFFF";
    cnt.fillRect(0, 0, 100, 100);
    cnt.fill();

    cnt.fillStyle = "#000000";

    cnt.beginPath();
    cnt.arc(25, 25, 10, 0, Math.PI*2, false);
    cnt.closePath();
    cnt.fill();

    cnt.beginPath();
    cnt.arc(50, 50, 10, 0, Math.PI*2, false);
    cnt.closePath();
    cnt.fill();

    cnt.beginPath();
    cnt.arc(75, 75, 10, 0, Math.PI*2, false);
    cnt.closePath();
    cnt.fill();
}
</script>
</head>
<body onload="init()">
<canvas id="canvas" width="600" height="400"></canvas>
</body>
</html>
```

キャンバスを指定します

コンテキストを取得します

影の線と色を指定します

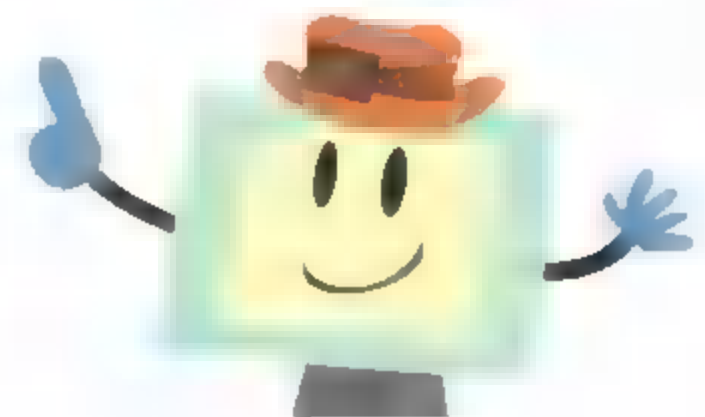
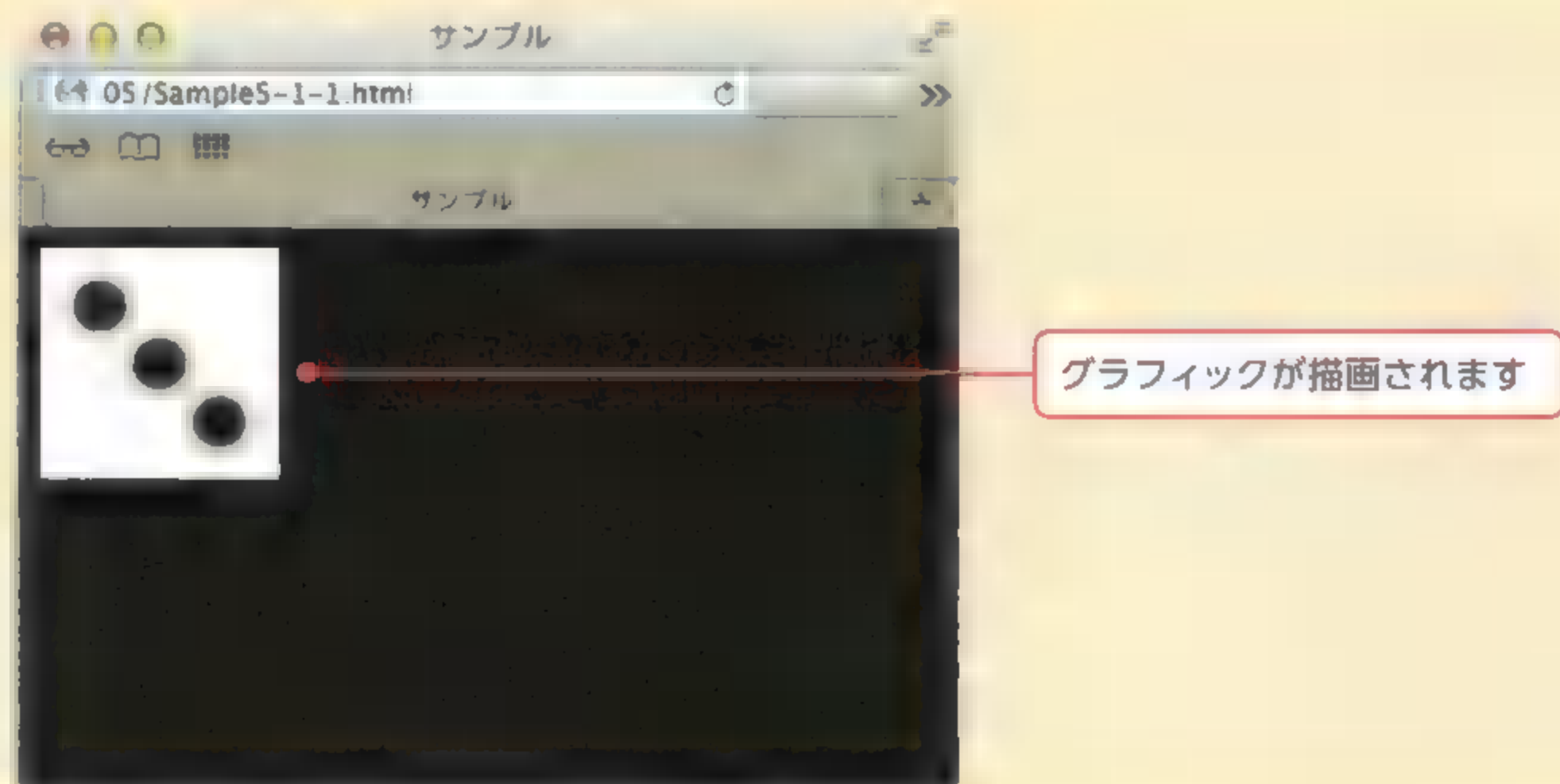
線と色を指定します

四角形を描画します

円を描画します

円を描画します

円を描画します

このキャンバスに描画されます

■Sample5-1-1の表示

大きな白く塗られた四角の中に、小さな黒い円を３つ描いています。サイコロの図として描画することができました。

キャンバスを使いこなす

それでは、キャンバスを使いこなし、さまざまなグラフィックを描いてみましょう。

円をランダムに描画する

まず、たくさんの円をランダムな位置・色で表示してみましょう。円を300個描画することにします。

Sample5-1-2.html ■ 円をランダムに描画する

```
<!DOCTYPE html>
<html lang="ja">
<head>
<link rel="stylesheet" href="Canvas.css">
<title>サンプル</title>
<script type="text/javascript">
function init()
{
   var c = document.getElementById("canvas");
   var cnt = c.getContext("2d");

   cnt.shadowBlur = 10;
   cnt.shadowColor = "#000000";
```

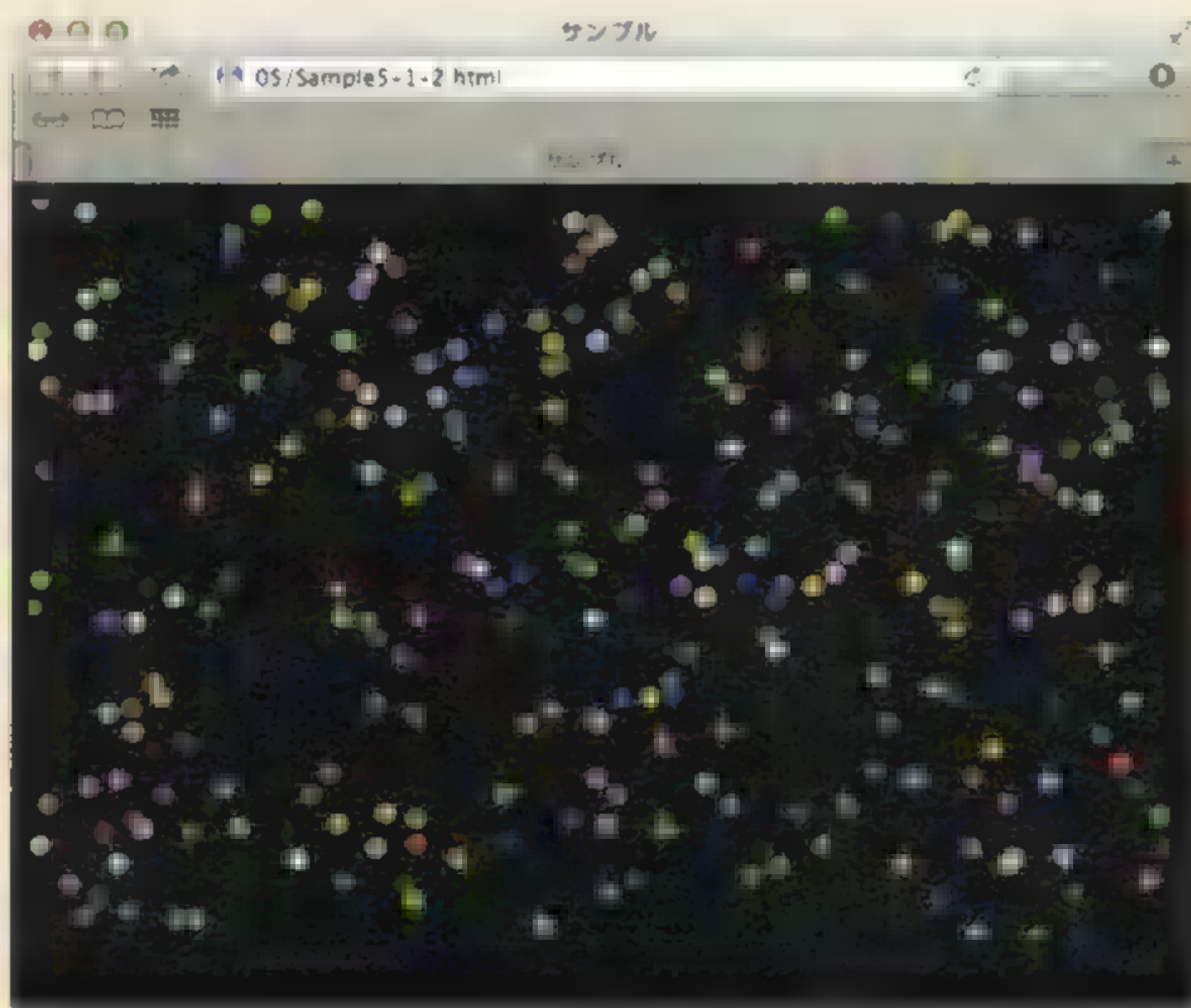
```
    for(var i=0; i<300; i++){

      var x = parseInt(Math.random() * 600);
      var y = parseInt(Math.random() * 400);
      var r = parseInt(Math.random() * 255);
      var g = parseInt(Math.random() * 255);
      var b = parseInt(Math.random() * 255);

      cnt.beginPath();
      cnt.arc(x, y, 5, 0, Math.PI*2, false);
      cnt.closePath();
      var color = "rgb(" + r + "," + g + "," + b + ")";

      cnt.fillStyle = color;
      cnt.fill();
    }
  }
</script>
</head>
<body onload="init()">
<canvas id="canvas" width="600" height="400"></canvas>
</body>
</html>
```

円を300個描画します

x座標を作成します

y座標を作成します

色を作成します

■ Sample5-1-2の表示

ここでは、繰り返し文を使って300個の円を描画しています。描画位置と色をランダムな値を使うことにします。描画位置は600×400のキャンバスの範囲におさまるように、次のようにしてランダムな位置を得ることにします。

```
var x = parseInt(Math.random() * 600);   // x座標を作成します
var y = parseInt(Math.random() * 400);   // y座標を作成します
```

色は、次のようにしてランダムな色としています。

```
var r = parseInt(Math.random() * 255);   // 赤を作成します
var g = parseInt(Math.random() * 255);   // 緑を作成します
var b = parseInt(Math.random() * 255);   // 青を作成します
...
var color = "rgb(" + r + "," + g + "," + b + ")";   // 色を作成します
```

色はrgb()関数を使って、次のような書式で指定します。

構文

```
"rgb(赤成分, 緑成分, 青成分)"
```

ここでは各成分をランダムな値とし、それぞれの値を変数r、g、bに記憶しています。シングルクォーテーションとダブルクォーテーションが、正しく対応しているかどうか注意して入力してください。

rgb(赤成分，緑成分，青成分)の各成分は、0～255の数値であらわします。rgb(0,0,0)は黒、rgb(255,255,255)は白をあらわします。

画像を描画する

今度はキャンバスに画像を描画してみましょう。キャンバスの機能を使って、画像の描画位置を指定したり、半透明にしたりすることができます。

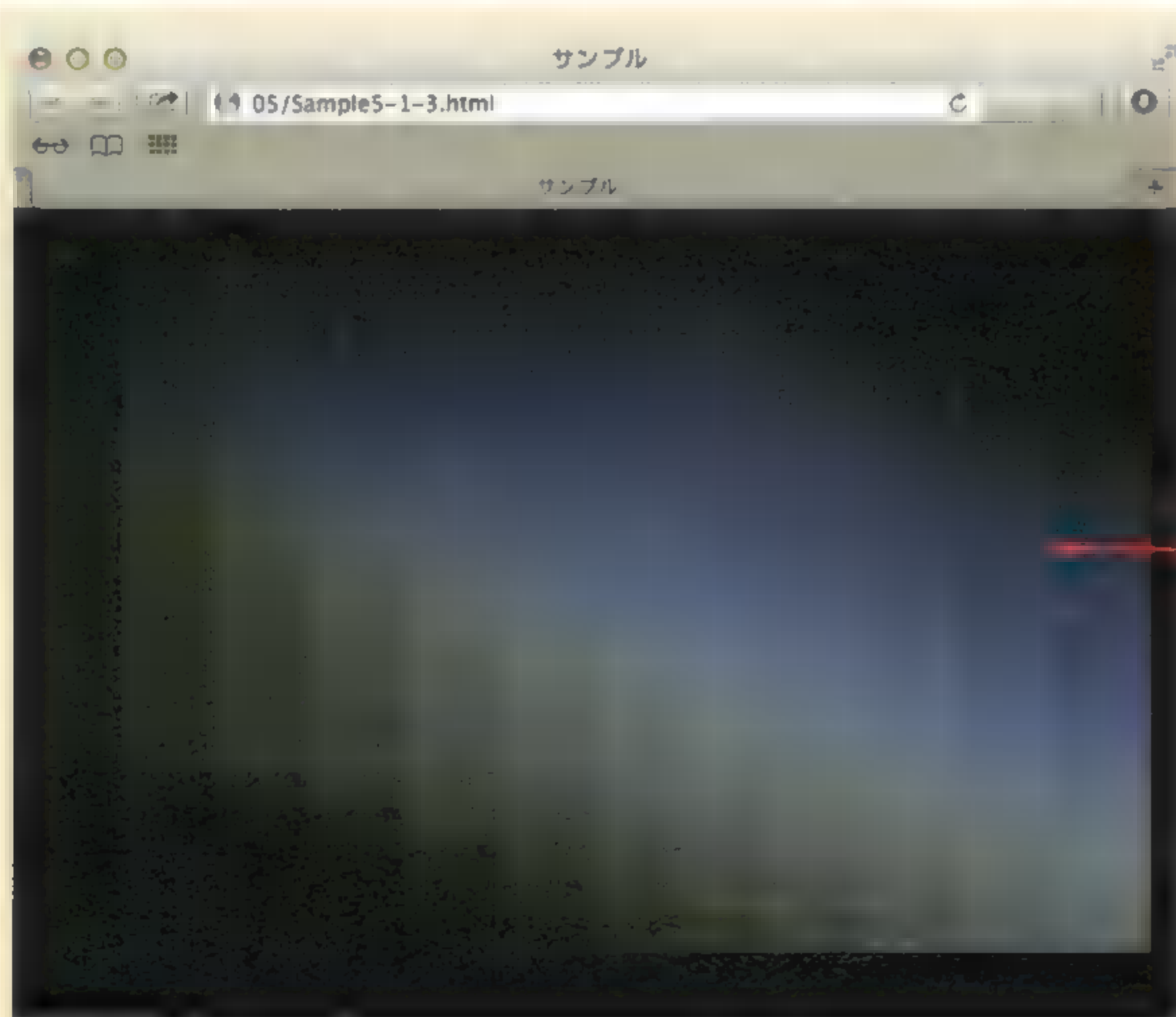
Sample5-1-3.html ■ 画像を描画する

```
<!DOCTYPE html>
<html lang="ja">
```

```
<head>
<link rel="stylesheet" href="Canvas.css">
<title>サンプル</title>
<script type="text/javascript">
var img = new Image();
img.src = "pic.jpg";

function init()
{
   var c = document.getElementById("canvas");
   var cnt = c.getContext("2d");

   cnt.globalAlpha = 0.2;
   for(var i=0; i<200; i=i+10){
      cnt.drawImage(img, i*5, i*2);
   }
}
</script>
</head>
<body onload="init()">
<canvas id="canvas" width="600" height="400"></canvas>
</body>
</html>
```

透明度を指定します（cnt.globalAlpha = 0.2;）

描画位置を変更します（cnt.drawImage(img, i*5, i*2);）

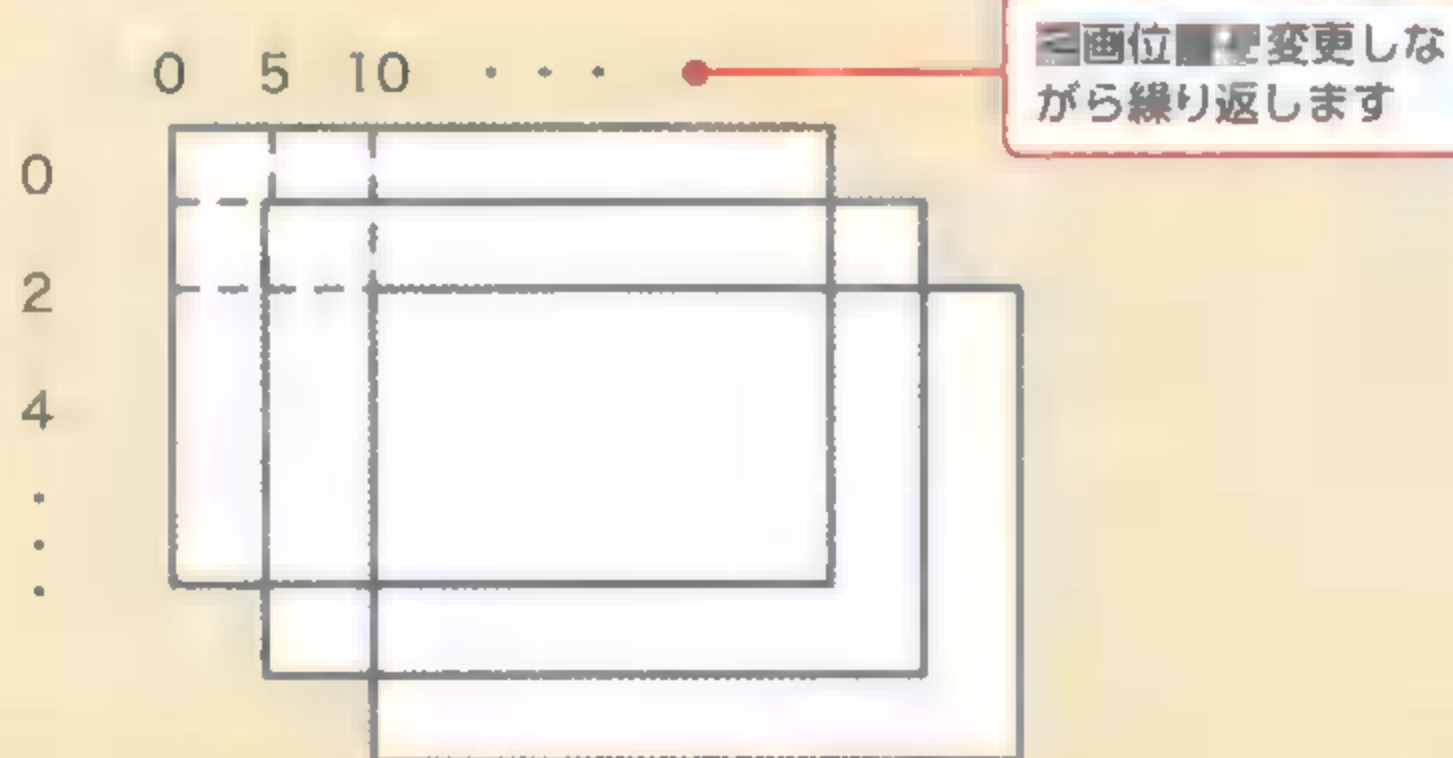
■ Sample5-1-3の表示

画像の透明度はコンテキストオブジェクトのglobalAlphaプロパティによって指定します。

```
cnt.globalAlpha = 0.2;
```

透明度を指定します

画像の透明度は小数で指定します。なお、プログラミングの世界では、小数と整数の区別をつけることが一般的です。透明度0を指定する場合には、通常0.0と指定します。

画像はdrawImage()メソッドで描画します。イメージオブジェクト、x座標、y座標の順で指定します。ここでは描画位置のx座標を5、y座標を2ずつずらしながら、繰り返し描画するようにしています。

■描画位置の変更

描画のためのメンバ

Column

この節のプログラムでは、2dコンテキストの描画のための[illegible]を使いました。2dコンテキストには、このほかにも次のようなメンバがあります。

■2dコンテキストの主なメンバ

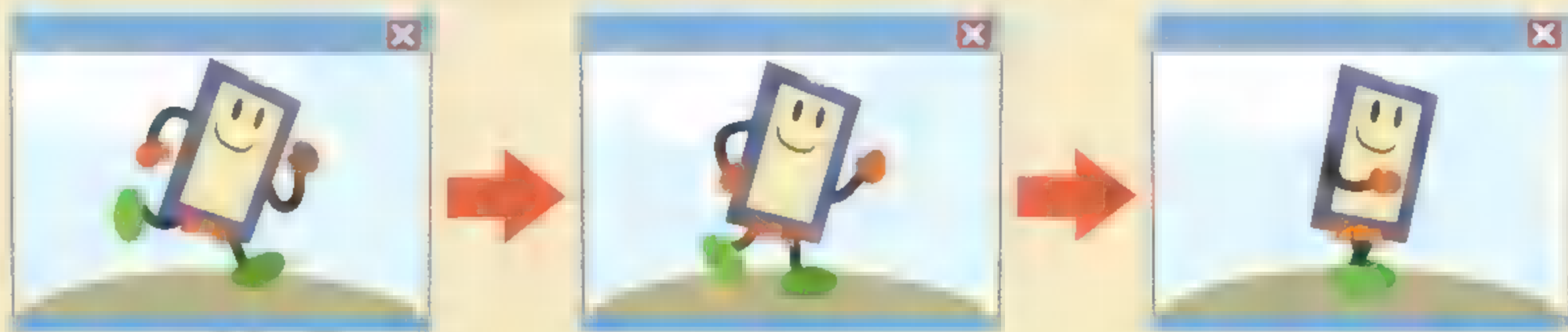
[illegible]	[illegible]
beginPath()	パスを開始する
closePath()	パスを閉じる
moveTo(x, y)	移動する
lineTo(x, y)	指定位置まで線を引く
arc(x, y, r, s, e, c)	(x, y)を左上端とする半径r、角度s〜eの円弧を描く (c=true/false：左回り／右回り)
arcTo(x1, y1, x2, y2, r)	(x1, y1)と(x2, y2)を半径rの円弧でつなぐ
stroke()	[illegible]で描画する
fill()	塗りつぶしで描画する
fillRect(x, y, w, h)	(x, y)を左上端とする[illegible]w高さhの塗りつぶし四角形を描く
storkeRect(x, y, w, h)	(x, y)を左上端とする[illegible]w高さhの四角形を描く
clearRect(x, y, w, h)	(x, y)を左上端とする幅w高さhの四角形を削除する
strokeText(txt, x, y)	(x, y)から線テキストを描く
fillText(txt, x, y)	(x, y)から塗りつぶしテキストを描く
clip()	クリッピングする
createLinearGradient(x0, y0, x1, y1)	(x0, y0)から(x1, y1)にグラデーション
createRadialGradient(x0, y0, r0, x1, y1, r1)	(x0, y0, r0)から(x1, y1, r1)にグラデーション
drawImage(img, x, y)	(x, y)から画像を描画する
drawImage(img, x, y, w, h)	(x, y)から幅w高さhで画像を描画する
createPattern(img, repeat)	パターンを作る
scale(x, y)	変形する
strokestyle	線色
lineWidth	線の幅
lineCap	[illegible]の端（butt/ round/square)
fillStyle	塗りつぶし色
shadowBlur	影
shadowColor	影色
shadowOffsetX	X方向の影幅
shadowOffsetY	Y方向の影高さ

5-2 アニメーションを作成しよう

チャレンジ! アニメーションをしよう

キャンバスを使えば、グラフィカルなアニメーションを作成することができます。アニメーションでは、画像や図形を一定時間ごとに描画することで動きをつけています。

こうしたアニメーションを作成するためには、キャンバスとタイマーを組み合わせるとよいでしょう。この節では、キャンバスを使ったアニメーションを作成してみましょう。

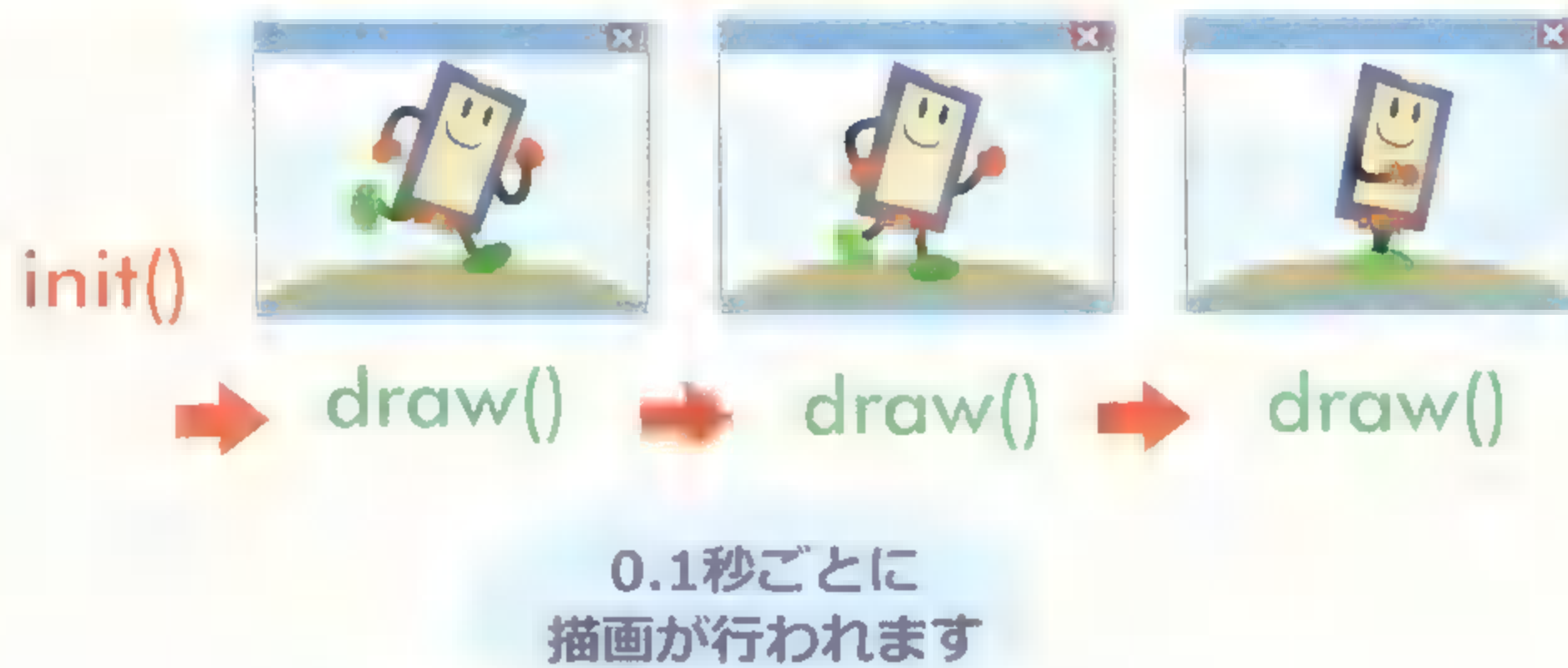
■アニメーション

キャンバスとタイマーでアニメーション

アニメーションを行うためには、キャンバスとタイマーを組み合わせて使います。

たとえば、0.1秒ごとに処理をするには、draw()という処理を次のように指定します。

```
function init()
{
   var t = window.setInterval("draw()", 100);   ← 0.1秒刻みにdraw()を処理します
}
function draw()
{
   ...                                          ← グラフィックを描画するようにします
}
```

draw()の中で、図形・画像の描画位置や透明度などが変更されるようにしておくと、タイマーによって0.1秒ごとに処理が行われますので、動きのあるアニメーションを実現することができます。

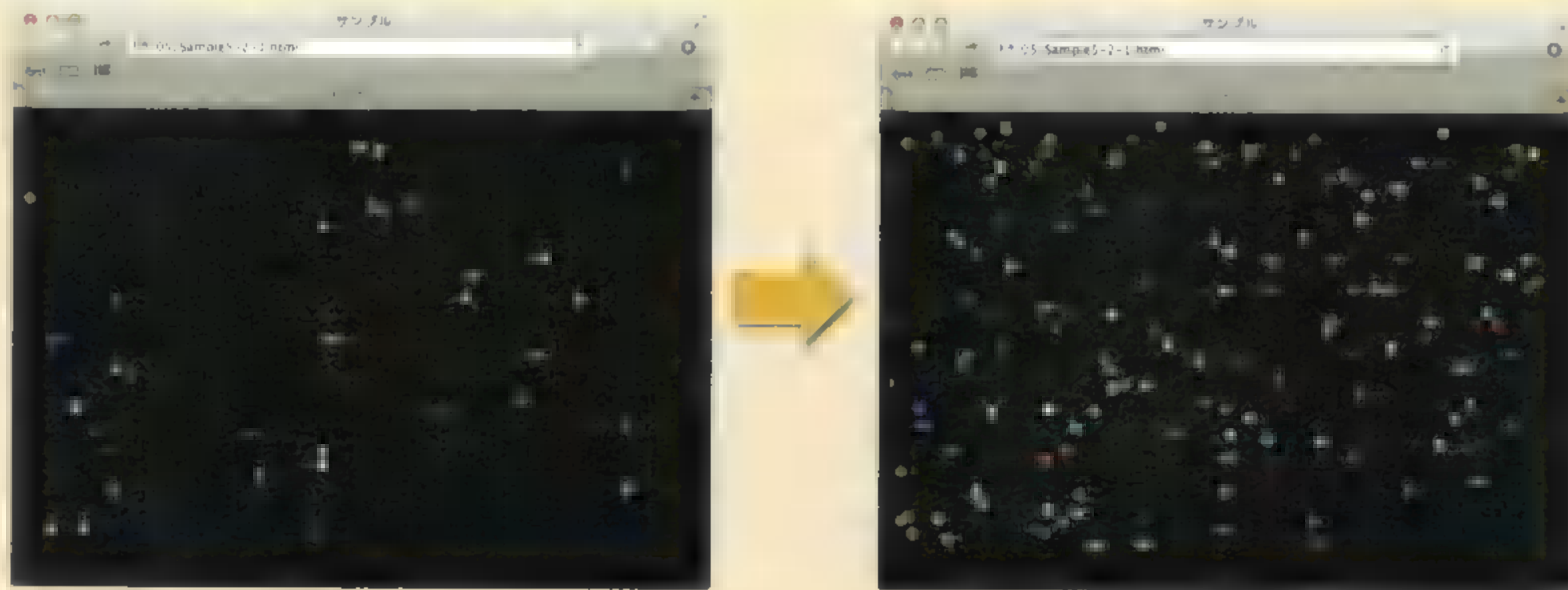
■連続した描画でアニメーションを実現

アニメーションを作成する

それでは、実際にアニメーションをしてみましょう。円をランダムに描画することにします。たくさんの円を描くプログラムはすでに作成しました。今度は0.1秒ごとに1個ずつ描画するアニメーションとしてみます。

Sample5-2-1.html ■ アニメーションする

```
<!DOCTYPE html>
<html lang="ja">
<head>
<link rel="stylesheet" href="Canvas.css">
<title>サンプル</title>
<script type="text/javascript">
function init()
{
   var t = window.setInterval("draw()", 100);   ← 0.1秒ごとに描画します
}
function draw()
{
   var c = document.getElementById("canvas");
   var cnt = c.getContext("2d");

   cnt.shadowBlur = 10;
   cnt.shadowColor = "#000000";
```

```
        var x = parseInt(Math.random() * 600);
        var y = parseInt(Math.random() * 400);
        var r = parseInt(Math.random() * 255);
        var g = parseInt(Math.random() * 255);
        var b = parseInt(Math.random() * 255);


        cnt.beginPath();
        cnt.arc(x, y, 5, 0, Math.PI*2, false);
        cnt.closePath();
        var color = "rgb(" + r + "," + g + "," + b + ")";

        cnt.fillStyle = color;
        cnt.fill();          ← 0.1秒ごとに1つの円を描画します
    }
</script>
</head>
<body onload="init()">
<canvas id="canvas" width="600" height="400"></canvas>
</body>
</html>
```

■ Sample5-2-1 の表示

このプログラムでは、0.1秒ごとに1つずつ円を描画していきます。たくさんの円を描画するプログラムと、ほとんど同じ方法で作成できることがわかるでしょう。繰り返し文のかわりに、タイマーを使って円を描いているわけです。

応用！ 画像によるアニメーション

キャンバスとタイマーを使えば、さまざまなアニメーションを作成することができます。今度は画像を使ったアニメーションをしてみましょう。

Sample5-2-2.html ■ 画像でアニメーションする

```
<!DOCTYPE html>
<html lang="ja">
<head>
<link rel="stylesheet" href="Canvas.css">
<title>サンプル</title>
<script type="text/javascript">
var img = new Image();
var i = 1.0;
function init()
{
   img.src="pic.jpg";
   var t = window.setInterval("draw()", 300);
}
function draw()
{
   var c = document.getElementById("canvas");
   var w = c.width;
   var h = c.height;
   var cnt = c.getContext("2d");

   cnt.clearRect(0, 0, w, h);
   if(i< 0.0) i=1.0;
   i = i - 0.1;
   cnt.globalAlpha = i;         ← 透明度を変更します

   cnt.drawImage(img, 0, 0);    ← 画像を描画します
}
</script>
</head>
<body onload="init()">
<canvas id="canvas" width="600" height="400"></canvas>
</body>
</html>
```

■ Sample5-2-2の表示

ここでは、タイマーで時間が経過するたびに、画像の透明度を変更しています。この結果、徐々に薄くなる画像のアニメーションとなっています。

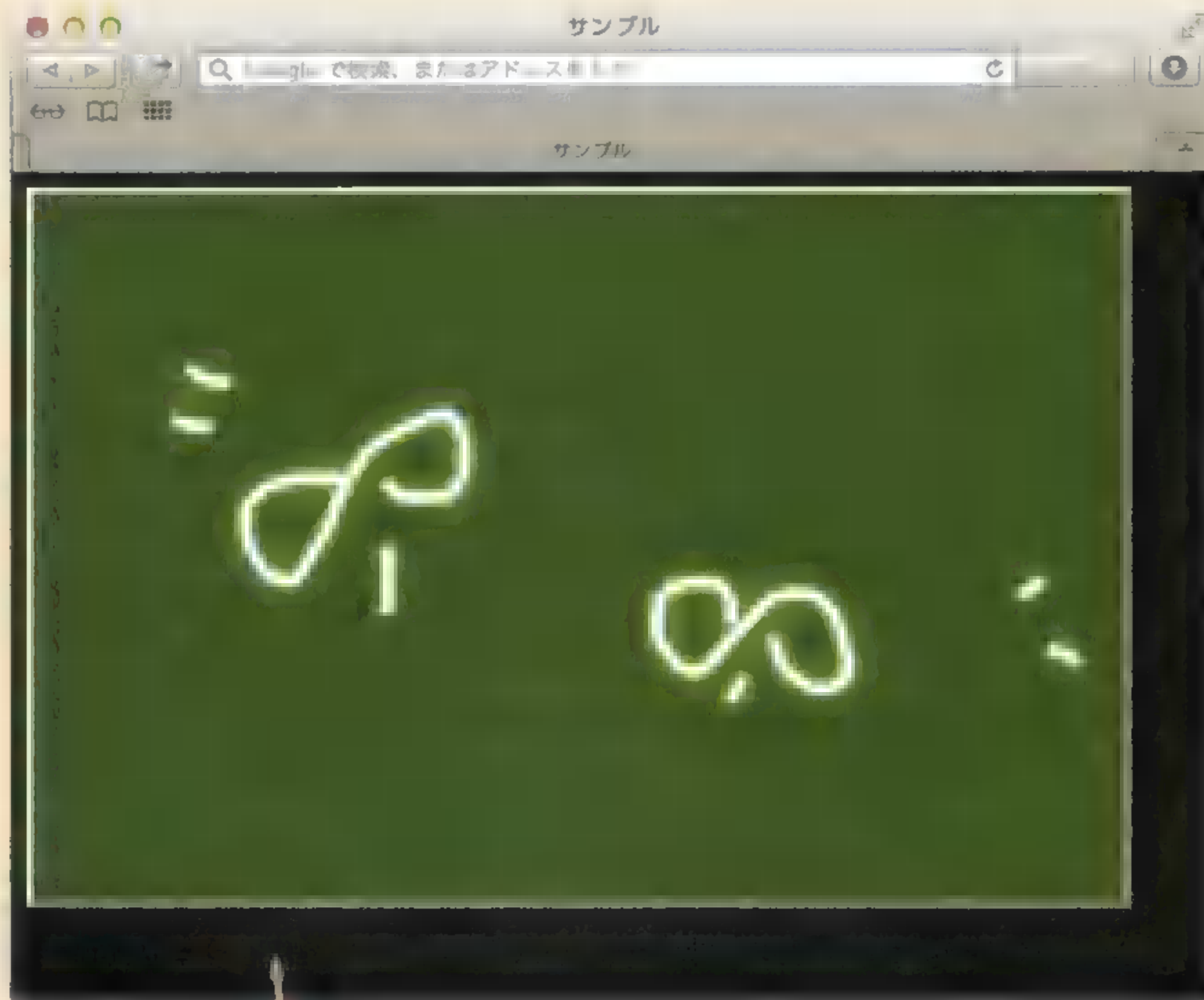
```
cnt.clearRect(0, 0, w, h);
if(i< 0.0) i=1.0;
i = i - 0.1;
cnt.globalAlpha = i;
```

- `if(i< 0.0) i=1.0;` ― 透明度が0.0より低くなったら元に戻します
- `cnt.globalAlpha = i;` ― [illegible]の透明度を変更します

5-3 実践！ペイントアプリを作ろう

チャレンジ！ ペイントアプリを作成する

この章の最後に、Webページにキャンバスを配置し、マウスを動かして絵を描くWebアプリを作成してみましょう。これまでの知識を活用すれば、実用的なペイントアプリも作成することができるのです。

■ペイントアプリ

Step 1 ペイントアプリの画面を設計する

まず、アプリの画面を考えましょう。このアプリはキャンバスを黒板にみたて、黒板のようなレイアウトを設計することにします。黒板の色・形は、キャンバス要素のレイアウトとしてスタイルシートで指定することにします。

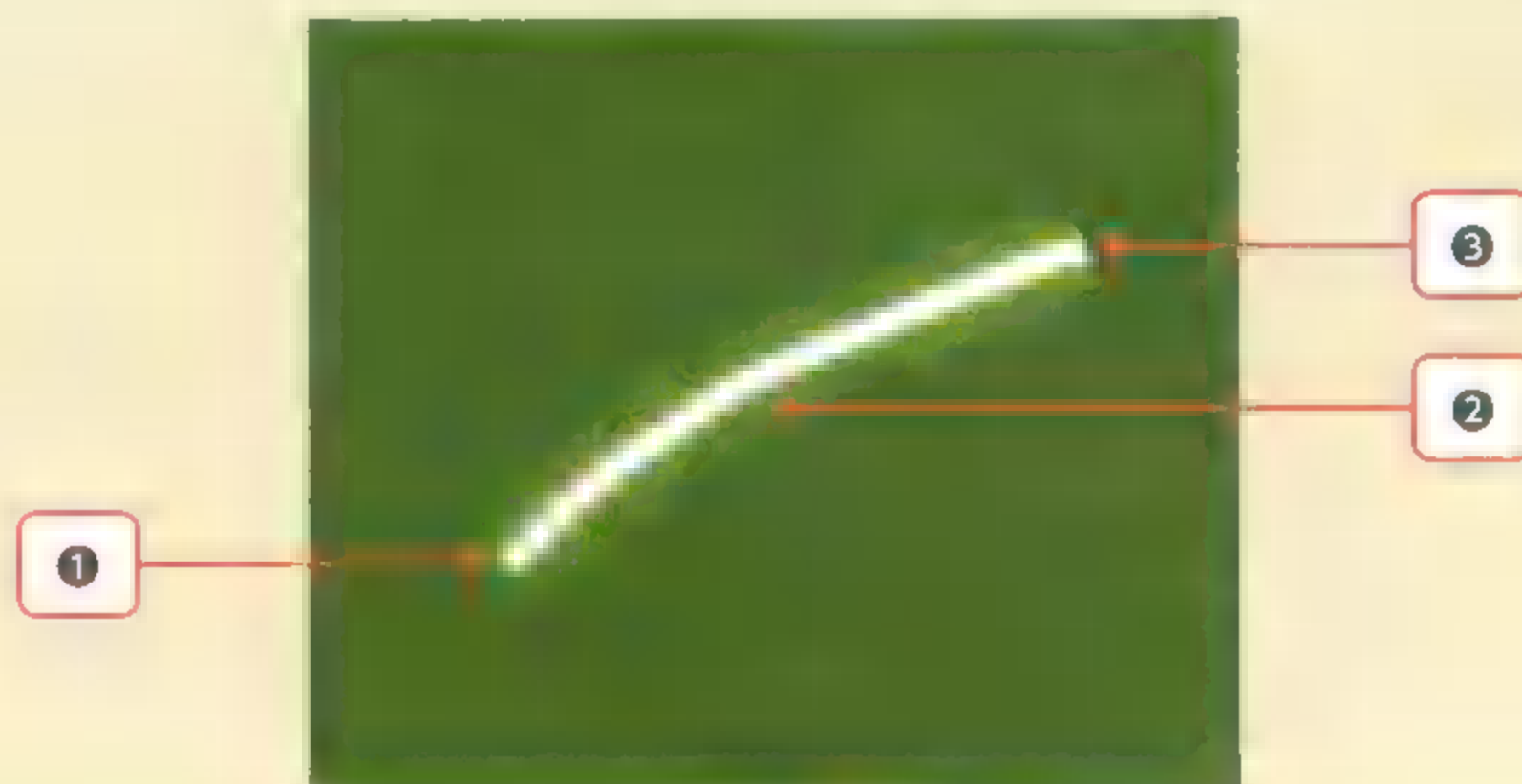
Boarrd.css ■ ペイントアプリのスタイルシート

```
body{
   background-color: #000000;
}
canvas{
   background-color: #006400;   ← キャンバスの背景色を緑に設定します
   border-style: outset;
   border-color: #FFFFFF;   ← キャンバスの境界色を白に設定します
}
```

Step 2 マウスでペイントするには？

ペイントアプリでは、ユーザーがマウスで操作したときの処理を設計する必要があります。マウスで操作したときの処理は、次のようにすればよいでしょう。

■ マウス操作したときの処理

操作	処理内容
❶マウスを押したとき	マウスによる描画を開始したことを設定し、開始位置を記する
❷マウスを動かしたとき	位置を取得して描画を行う
❸マウスを離したとき	マウスによる描画が終了したことを設定する

マウスを押したときに描画を開始したことを設定し、開始位置を記録します（❶）。次に、マウスを動かしたときに現在位置を調べ、開始位置から現在位置までの描画を行います（❷）。そして、マウスを離したときに描画が終了したことを設定するのです（❸）。

Step3 マウスの処理を作成する

処理内容が決まったら、マウスの処理を記述していきましょう。

キャンバス上のマウスの動きを受け取る

キャンバスがマウスの動きに反応するようにするためには、まず次のように、キャンバス上のマウスの動作を受け取るように設定しておくことが必要です。次のコードでは、❶でマウスを押したとき、❷でマウスを動かしたとき、❸でマウスを離したときの設定を行っています。

この方法は、第3章で紹介した、ユーザーがWebページを操作したときの処理方法とは異なっているので注意してください。

```
function init()
{
    var c = document.getElementById("canvas");
    c.addEventListener("mousedown", begin_draw, false);  ❶
    c.addEventListener("mousemove", draw, false);  ❷
    c.addEventListener("mouseup", end_draw, false);  ❸
}
```

キャンバス上でのマウスの動作も、第3章で紹介した方法と考え方は同じです。ただし、キャンバスはHTML5の要素であるため、イベントを受け取る方法が第3章の方法とは少し違っています。キャンバス上の操作に反応するようにするためには、addEventListener()を使います。

マウス操作の処理を書く

このあと、先ほど考えたマウスの操作時の処理を記述することにします。❶～❸の処理を記述していきましょう。

❶マウスを押したときの処理

```
function begin_draw(e)
{
   isDraw = true;
   bx = e.clientX;
   by = e.clientY;
}
```

「描画中」であることを設定します

描画位置を設定します

❷マウスを動かしたときの処理

```
function draw(e)
{
   if(isDraw == true){
     ...
   }
}
```

「描画中である」と設定されていれば描画します

❸マウスを離したときの処理

```
function end_draw()
{
   isDraw = false;
}
```

描画が終了したことを設定します

なお、true/falseは、2択をあらわすJavaScriptの値です。trueは「はい」、falseは「いいえ」という意味になります。

ここでは、trueを「描画中である」、falseを「描画中でない」という意味で使っています。そこで、アプリの開始時には、「false」を設定しておきましょう。描画位置の情報も、最初は(0,0)としておきます。

```
<script type="text/javascript">
var isDraw = false;
var bx = 0;
var by = 0;
...
```

最初は「描画中でない」を設定します

描画位置を(0,0)としておきます

描画中かそうでないかをあらわす変数isDrawは、すべての処理の外側に書く必要があります。マウスの1つの動作が終わったときにも、描画中であるかそうでないかを記憶しておく必要があるためです。描画位置bx、byも同じ理由で、処理の外側に書きます。

Eventオブジェクト

Eventオブジェクトのclient X、clientYプロパティによって、マウスの位置を調べることができます。Eventオブジェクトは、このほか次のようなプロパティを持っています。必要に応じて利用するとよいでしょう。

■Eventオブジェクトの主なプロパティ

プロパティ	説明
type	イベントの種類
clientX	ブラウザ上X座標
clientY	ブラウザ上Y座標
screenX	スクリーン上X座標
screenY	スクリーン上Y座標
keyCode	押されたキーコード

Step 4 ペイントアプリを完成させよう

それではペイントアプリを完成させましょう。黒板のようなキャンバスが表示され、マウスをクリックして動かすと絵を描くことができます。

Sample5-3-1.html ■ ペイントアプリ

```
<!DOCTYPE html>
<html lang="ja">
<head>
<link rel="stylesheet" href="Sample.css">
<title>サンプル</title>
<script type="text/javascript">
var isDraw = false;        初期設定を行います
var bx = 0;
var by = 0;

function init()
{
```

```
    var c = document.getElementById("canvas");
    c.addEventListener("mousedown", begin_draw, false);
    c.addEventListener("mousemove", draw, false);
    c.addEventListener("mouseup", end_draw, false);
}
function begin_draw(e)
{
    isDraw = true;
    bx = e.clientX;
    by = e.clientY;
}
function draw(e)
{
    if(isDraw == true){
        var c = document.getElementById("canvas");
        var cnt = c.getContext("2d");

        var x = e.clientX;
        var y = e.clientY;

        cnt.shadowBlur = 10;
        cnt.shadowColor = "#FFFFFF";
        cnt.lineWidth = 5;
        cnt.strokeStyle = "#FFFFFF";

        cnt.beginPath();
        cnt.moveTo(bx, by);
        cnt.lineTo(x, y);
        cnt.closePath();
        cnt.stroke();
        bx = x;
        by = y;
    }
}
function end_draw()
{
    isDraw = false;
}
</script>
</head>
<body onload="init()">
<canvas id="canvas" width="600" height="400"></canvas>
</body>
</html>
```

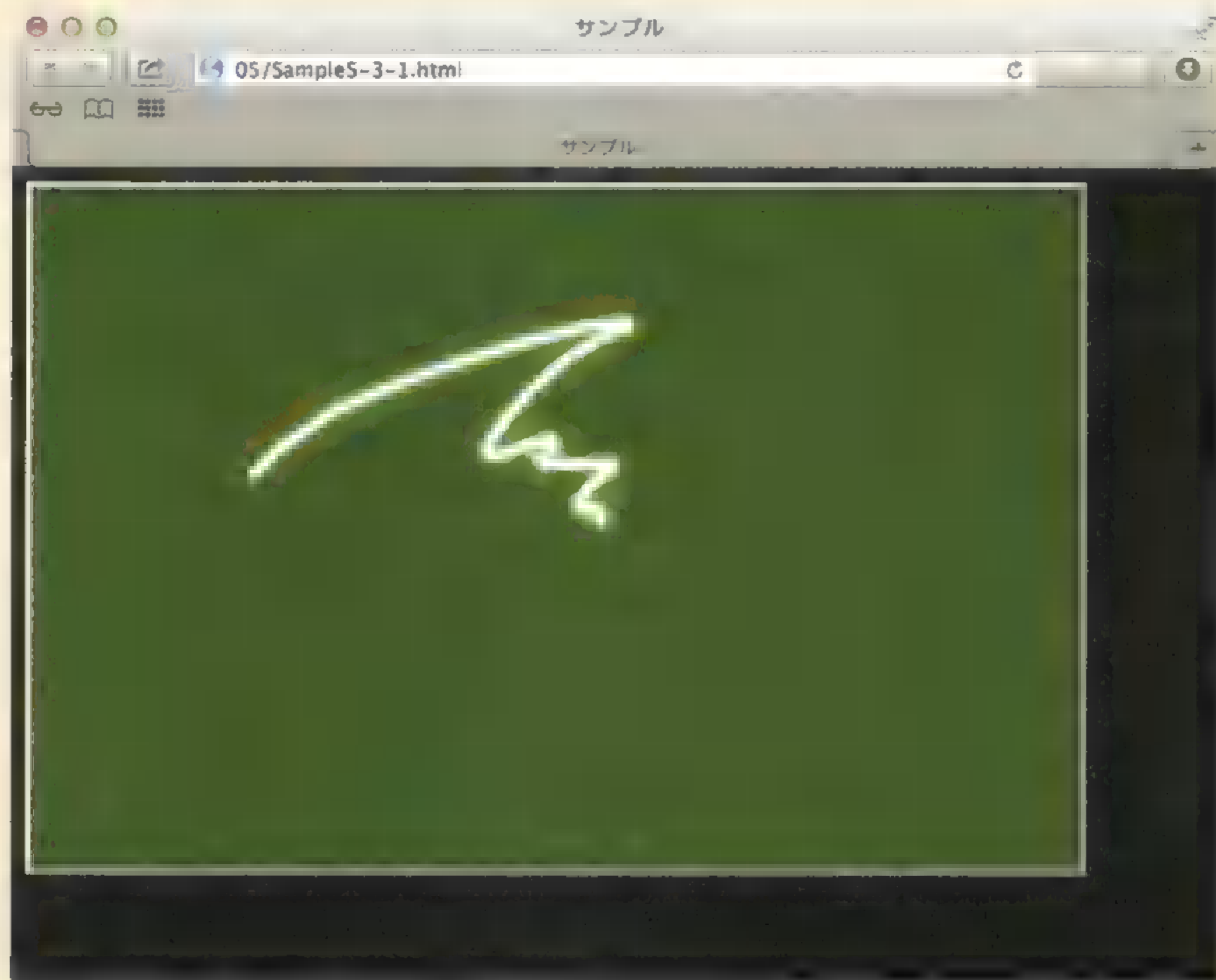
マウスの動作を受け取るように設定します

「描画中である」ことを設定します

描画位置を設定します

「描画中である」と設定されていれば描画します

描画が終了したことを設定します

■ Sample5-3-1の表示

アプリの機能を増やしていく

ここでは、黒板アプリとして、黒板に白チョークで描画する機能だけを作成してみました。

このほかにも、さまざまな機能を追加することが考えられるでしょう。黒板消しの機能、チョークの色を変更する機能など、アプリにたくさんの機能を■やしていくことが考えられます。JavaScriptとそのオブジェクトの機能を学ぶことで、こうした機能を実現していくことができるようになります。アプリの機能を設計し、JavaScriptで自由自在に機能を実現できるようになるとよいでしょう。

6-1 マップを利用しよう

チャレンジ！ マップを表示しよう

JavaScriptによって、マップ（地図）を利用するWebページを作成することができます。マップをWebページ上に表示したり、マウスでクリックした位置に印をあらわしたりすることができるのです。

また、緯度・経度情報を取得して住所を表示したり。住所から緯度・経度情報を調べることもできます。

開発するWebサイトでマップを活用できれば便利です。この章では、マップを利用したWebページを作成してみましょう。

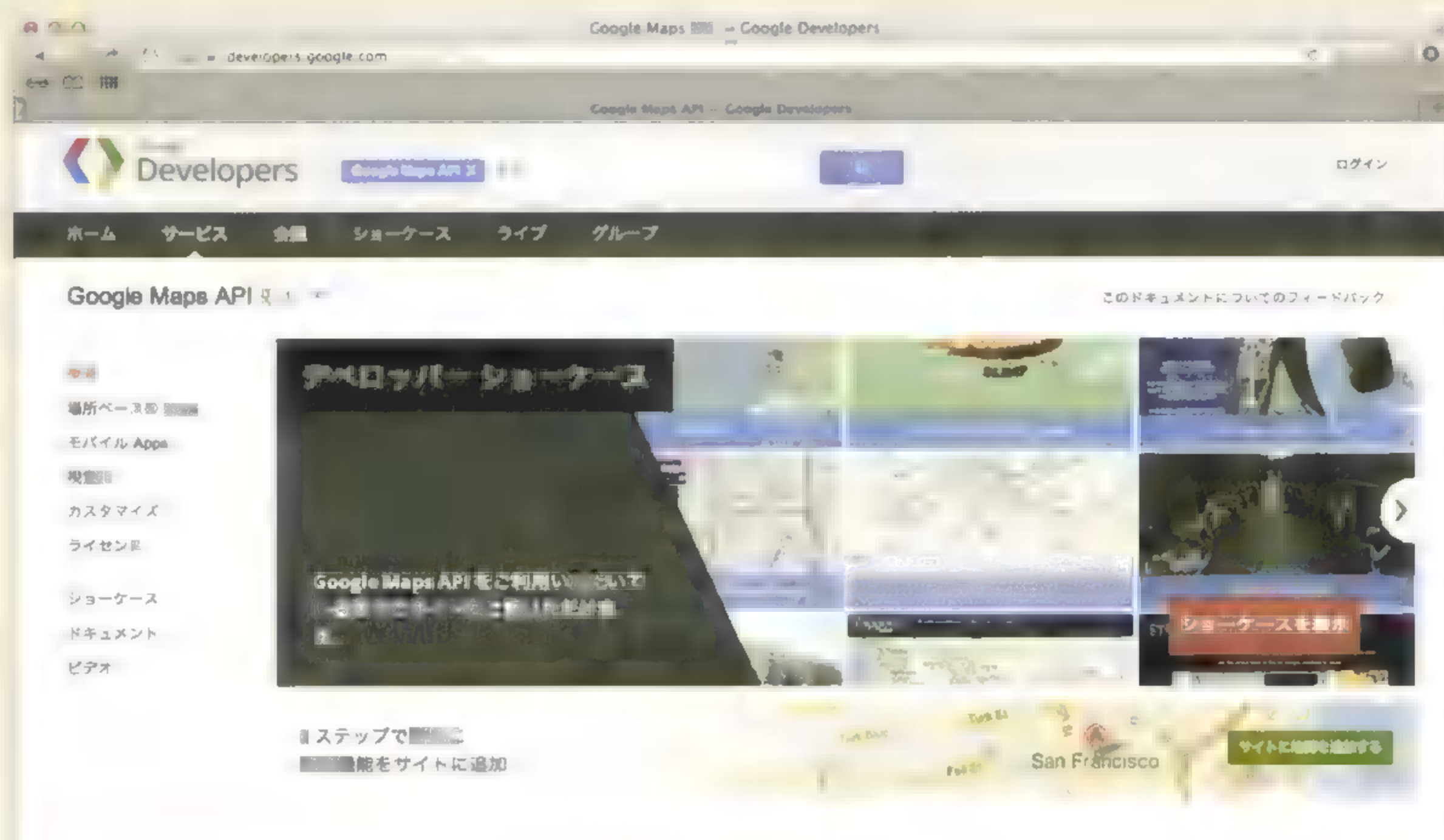
■ 地図を利用するページ

Googleマップとは

本書では、Google社が提供するGoogle Maps JavaScript API バージョン3（V3）を使用してマップを表示することにします。Googleマップに関する情報は、次のサイトから入手することができます。

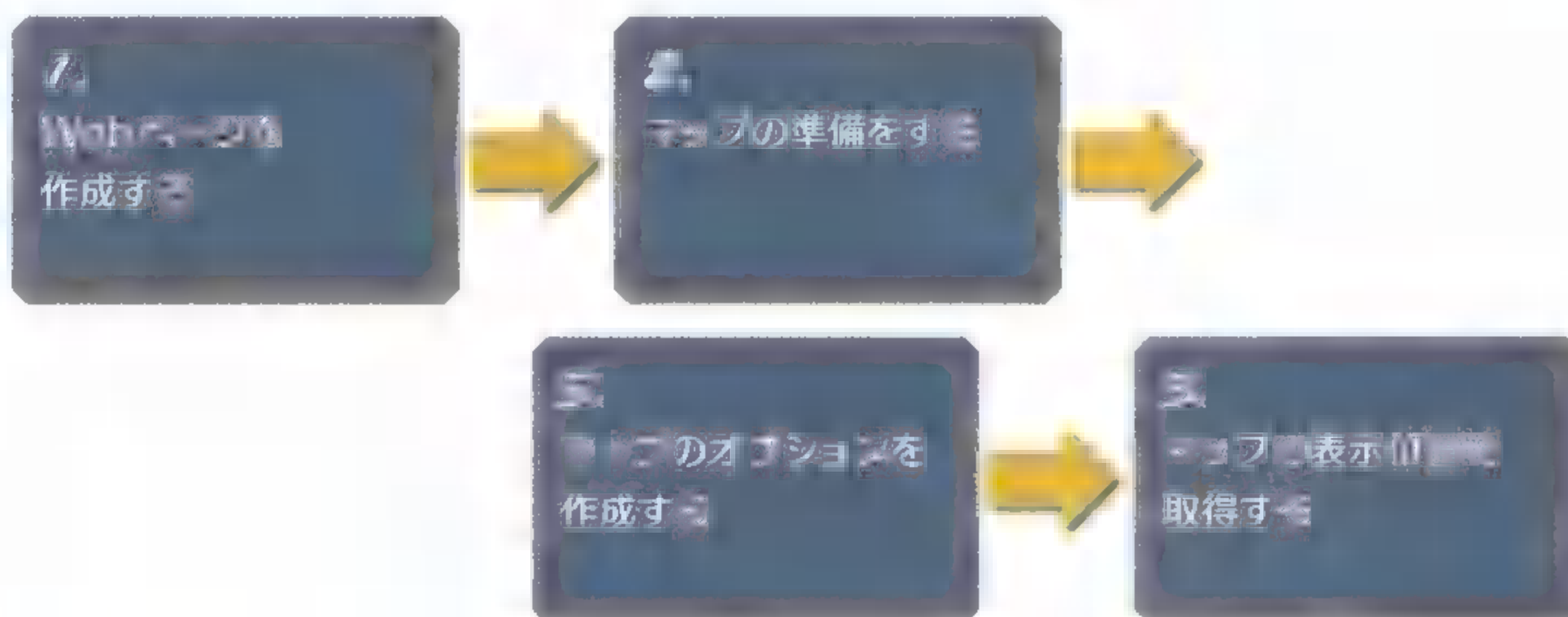
Google Maps JavaScript API V3

URL http://code.google.com/intl/ja/apis/maps/documentation/javascript/

■ Googleマップの情報ページ

Googleマップを利用するには、Google Maps APIキーを取得する必要があります。取得方法については、本書付録で説明していますので参考にしてみてください。

6-2 マップを準備しよう

マップの準備方法

まず最初に、Webページ上にマップを表示する方法を紹介しておきましょう。マップは次の方法で表示することにします。

Webページを作成する

まず、マップを表示するWebページの内容を考えておくことにしましょう。このとき、マップを表示するための位置を指定しておく必要があります。

マップを表示する範囲を「div」要素とし、ID名をつけることにします。ここではmymapという名前をつけることにしました。

```
<html>
...
<div id="mymap"></div>
</body>
</html>
```

また、Webページと一緒にマップの表示を行うCSSを作成します。このスタイルシートの中で、マップの幅・高さを指定してください。幅・高さを指定しないとマップが表示されません。そこでこの章では、次のスタイルシートを使うことにします。

Map.css ■ マップ用のスタイルシート

```
body{
    background-color: #000000;
}
h3{
    background-color: #000000;
    color: #FFFFFF;
}
#mymap{
    width: 600px;
    height: 400px;
}
```

マップの幅・高さを指定します

マップの準備をする

次に、マップを利用するために、HTML文書内に「script」要素を記述します。

この「script」要素は、ほかのJavaScriptのコードを記述した「script」要素とは別に記述する必要があるので注意してください。ここのXXXXXの部分には、Googleサイトで取得したAPIキーを指定してください。

```
<script type="text/javascript" src="http://maps.google.com/maps/api/js?key=XXXXX&sensor=
false">
</script>
```

APIキーの取得方法は、本書付録で紹介しています。参考にして取得・指定してみてください。

マップのオプションを作成する

それでは、新しい「script」要素に、マップを取り扱う処理を記述していくことにしましょう。

まず、マップを作成するために必須のオプションを記述します。次のコードをみてください。ここでは、マップのズームレベル（❶）、マップ中央の緯度・経度情報（❷）、マップタイプ（❸）を指定しています。

なお、緯度・経度情報を扱うオブジェクトは、new google.maps.LatLng(緯度,経度)で作成することができます。

```
var p =  new google.maps.LatLng(0,0);
var opt = {
   zoom: 2,
   center: p,
   mapTypeId: google.maps.MapTypeId.ROADMAP
};
```

[illegible]経度情[illegible]作成しています
❶ ズームレベルです
[illegible] マップの中央です
[illegible] マップの種類です

Tips

ここでは、北緯0度・東経0度という緯度・経度情報を作成し、マップの中央として指定しています。緯度経度を自由に指定することによって、表示される位置を変更することができるでしょう。たとえば、北緯36[illegible]・東経136[illegible]近辺の値を指定すると日本近辺の地図が表示されます。また、ズームレベルは、値が大きくなるほどズームして表示されます。

マップの表示位置を取得する

次に、マップを表示する要素をDOMを使って指定します（❹）。この要素に指定したオプションで地図を作成するのです（❺）。

```
var mm = document.getElementById("mymap");
var map = new google.maps.Map(mm, opt);
```

❹ マップを表示する[illegible]を取得します
❺ マップを作成します

これらの処理は、init()という名前の処理としてまとめ、「body」要素のonloadに指定しておきましょう。これで、HTML文書が読み込まれたときに、マップが作成・表示されることになります。

マップを表示する

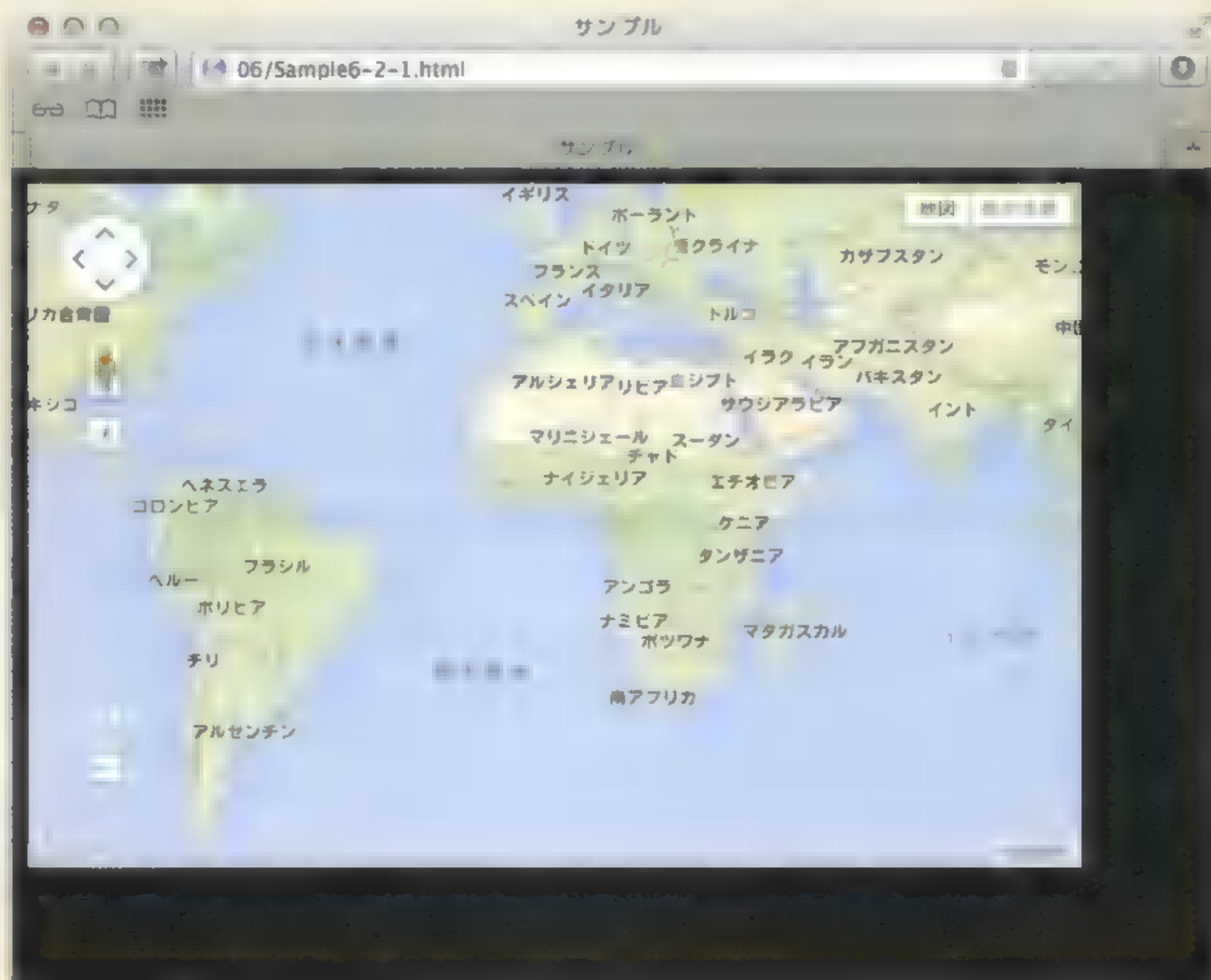
それでは、マップを表示するページを確認してみることにします。マップがWebページに表示されることを確認してみてください。

Sample6-2-1.html ■ マップを表示する

```
<!DOCTYPE html>
<html lang="ja">
<head>
<link rel="stylesheet" href="Map.css">
<title>サンプル</title>
```

```
<script type="text/javascript" src="http://maps.google.com/maps/api/js?key=XXXXX&sensor=
false"> ●───マップを利用します
</script>
<script type="text/javascript">
function init()
{
   var p =  new google.maps.LatLng(0,0);
   var opt = {
      zoom: 2, ●───❶ ズームレベルです
      center: p, ●───❷ マップの中心です
      mapTypeId: google.maps.MapTypeId.ROADMAP ●───❸ マップの種類です
   };
   var mm = document.getElementById("mymap"); ●───❹ マップを表示する要素を取得します
   var map = new google.maps.Map(mm, opt); ●───❺ マップを作成します
}
</script>
</head>
<body onload="init()">
<div id="mymap"></div>
</body>
</html>
```

■ Sample6-2-1 の表示

マップの[illegible]

マップタイプはマップの種類をあらわします。マップタイプには、次の種類（通常・航空写真・ハイブリッド・地形）があります。さまざまな表示を確認してみてください。

■ さまざまなマップタイプ

ROADMAP（通常）

SATELLITE（航空写真）

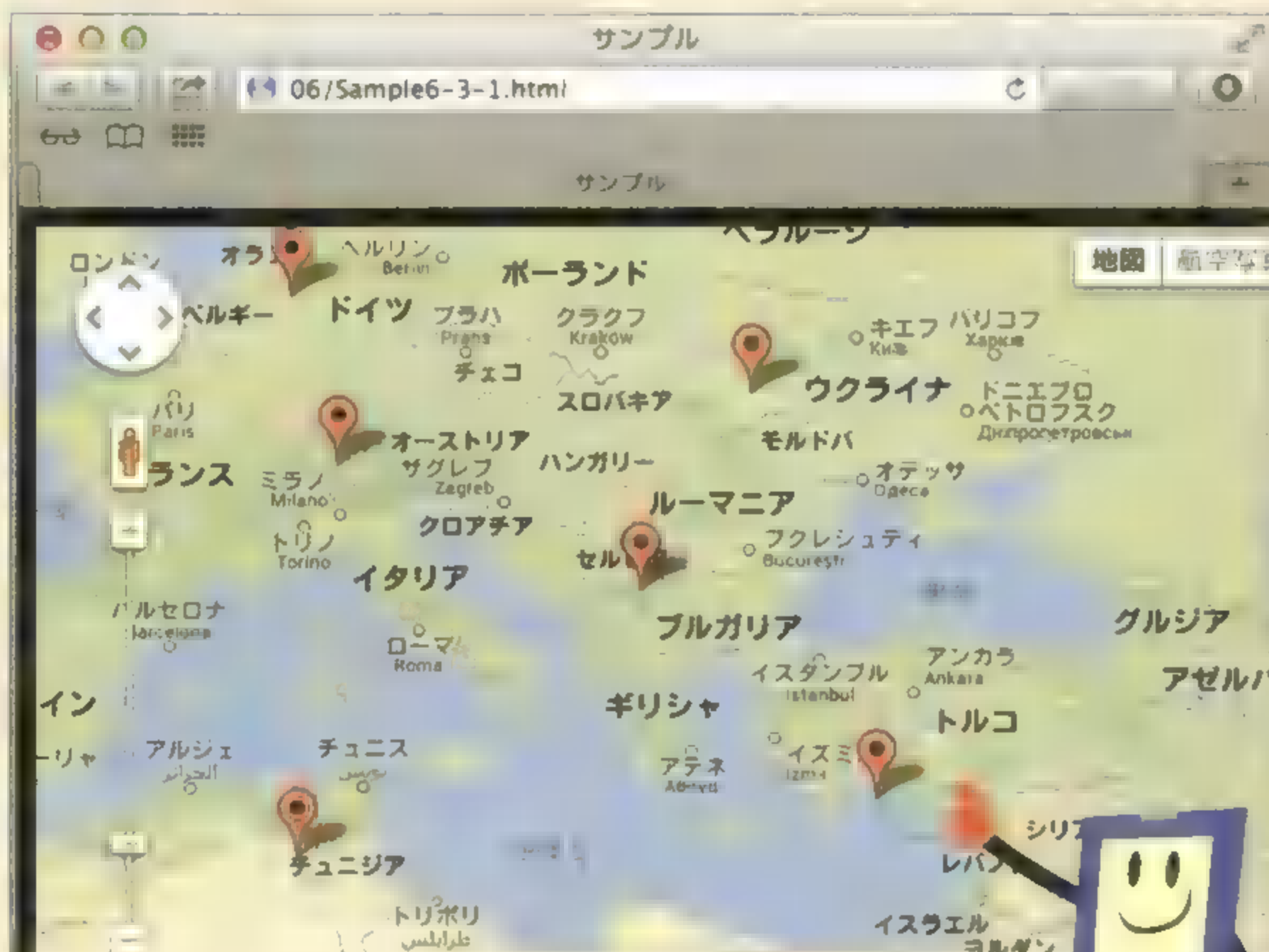
HYBRYD（ハイブリッド：航空写真と重ねる）

TERRAIN（地形）

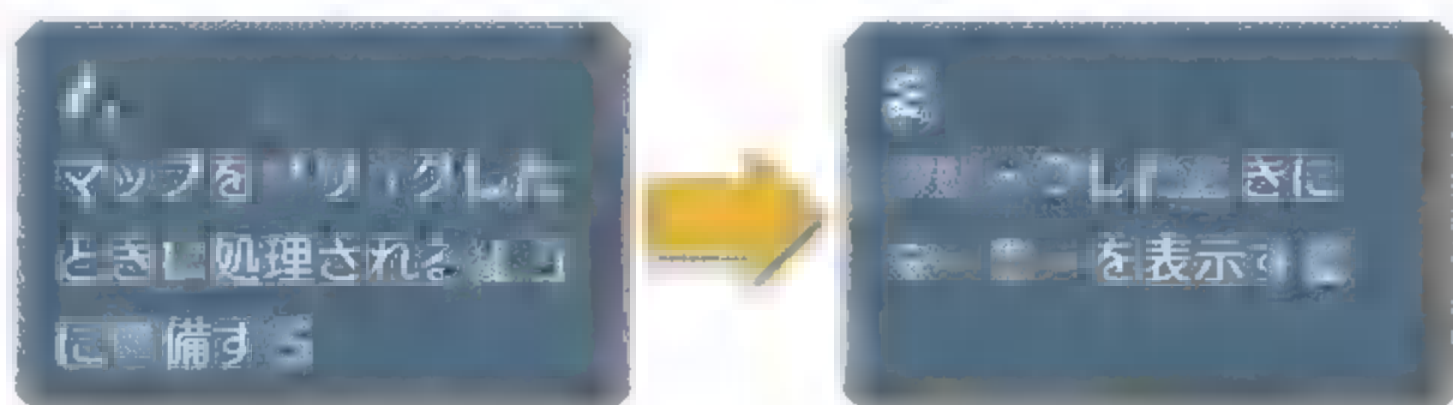
6-3 マーカーを表示してみよう

チャレンジ! マーカーを表示しよう

今度はマップをクリックしたときに、マーカーを表示するようにしましょう。マーカーとは、マップ上に表示できる印のことです。クリックした位置にマーカーを表示するようにするのです。

■ マーカーを使う

マーカーを
表示します

マーカーを表示するには？

クリックしたときにマーカーを表示するためには、次の手順が必要になります。

クリックを受け取る準備をする

まず、マップ上でクリック操作した場合に動作するようにするためには、次の方法で処理を登録する必要があります。

構文

```
google.maps.event.addListener(マップを表す変数名, "イベント", 処理名);
```

たとえば、変数mapで扱っているマップをクリックしたときに処理を行うには、次のように指定します。

```
google.maps.event.addListener(map, "click", map_click);

...
function map_click(e)
{
    ...
}
```

ここでは、map_clickという名前の処理を作成しておき、clickイベントが起こった場合に処理が行われるようにするのです。

マーカーを作成する

次に、マーカーに関するオプションを作成しておきます。ここでは、マーカーを表示する位置(❶)、マーカーを表示するマップをあらわす変数（❷)、マーカーのタイトル（❸）を指定しています。これらのオプションを指定して、マーカーを作成します（❹)。

```
var p = e.latLng;
   var mp = {
      position: p,
      map: map,
      title: "目的地"
   };

var mk = new google.maps.Marker(mp);
```

クリックした緯度・経度を調べます
❶マーカーを表示する位置と・・・
❷マーカーを表示するマップと・・・
❸マーカーのタイトルを指定し・・・
❹マーカーを作成します

マーカーを表示しよう

それでは、マーカーを表示するページを完成させてみましょう。

Sample6-3-1 ■ マーカーを表示する

```
<!DOCTYPE html>
<html lang="ja">
<head>
<link rel="stylesheet" href="Map.css">
<title>サンプル</title>
<script type="text/javascript" src="http://maps.google.com/maps/api/js?key=XXXXX&sensor=
false">
</script>
<script type="text/javascript">
var map;

function init()
{
   var p =  new google.maps.LatLng(0,0);
   var opt = {
      zoom: 3,
      center: p,
      mapTypeId: google.maps.MapTypeId.ROADMAP
   };
   var mm = document.getElementById("mymap");
   map = new google.maps.Map(mm, opt);

   google.maps.event.addListener(map, "click", map_click);
}
function map_click(e)
{
   var p = e.latLng;
   var mp = {
      position: p,
```

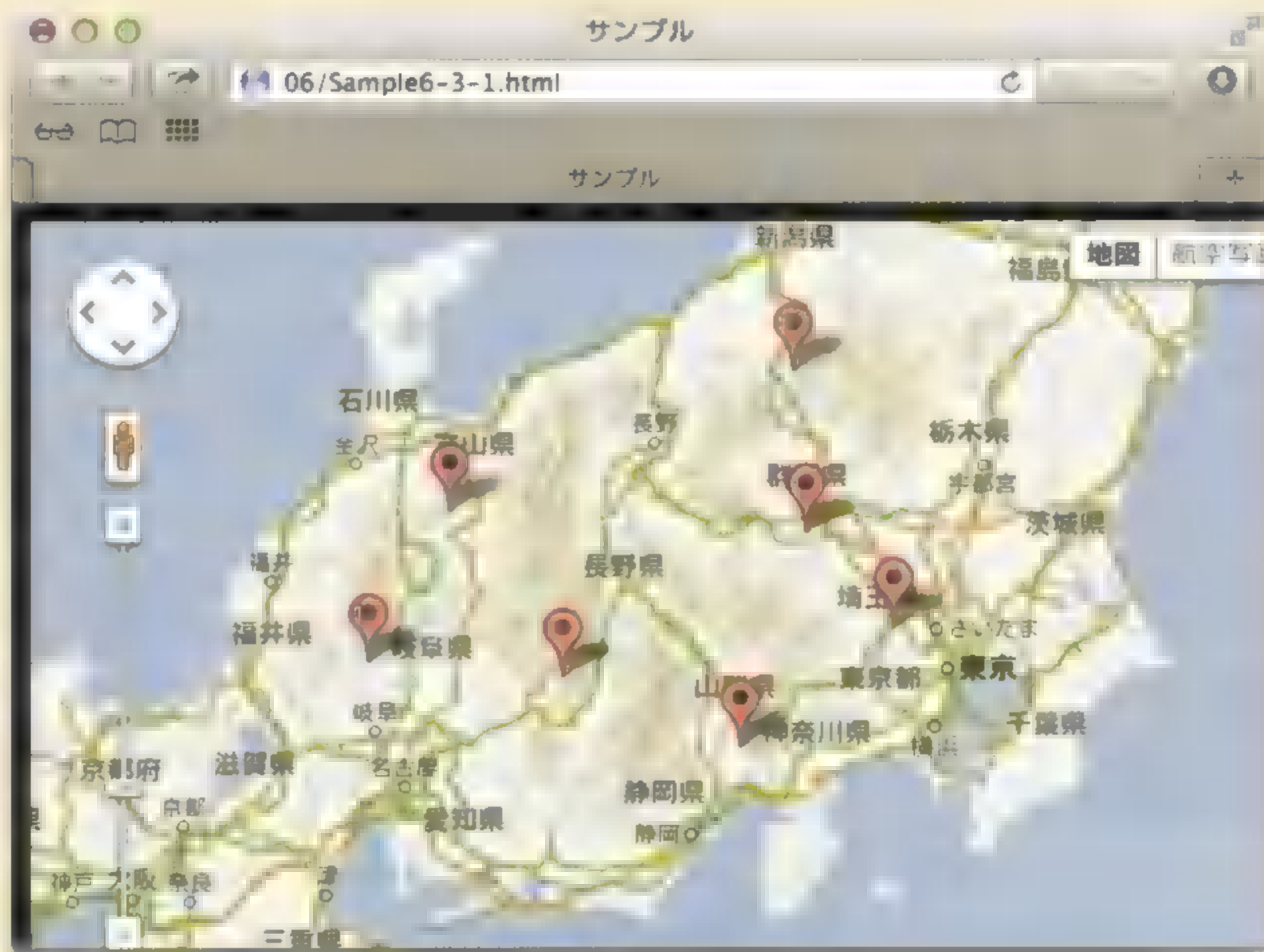
マップをクリックしたときに行われる処理を指定します

クリックした位置を調べて・・・

マップを活用しよう！

```
      map: map,
      title: "目的地"
    };
    var mk = new google.maps.Marker(mp);
}
</script>
</head>
<body onload="init()">
<div id="mymap"></div>
</body>
</html>
```

マーカーを作成します

■ Sample6-3-1の表示

マップ上のクリックした位置にマーカーを表示することができます。JavaScriptのプログラムによって、マップ上にマーカーを表示することができました。

Tips

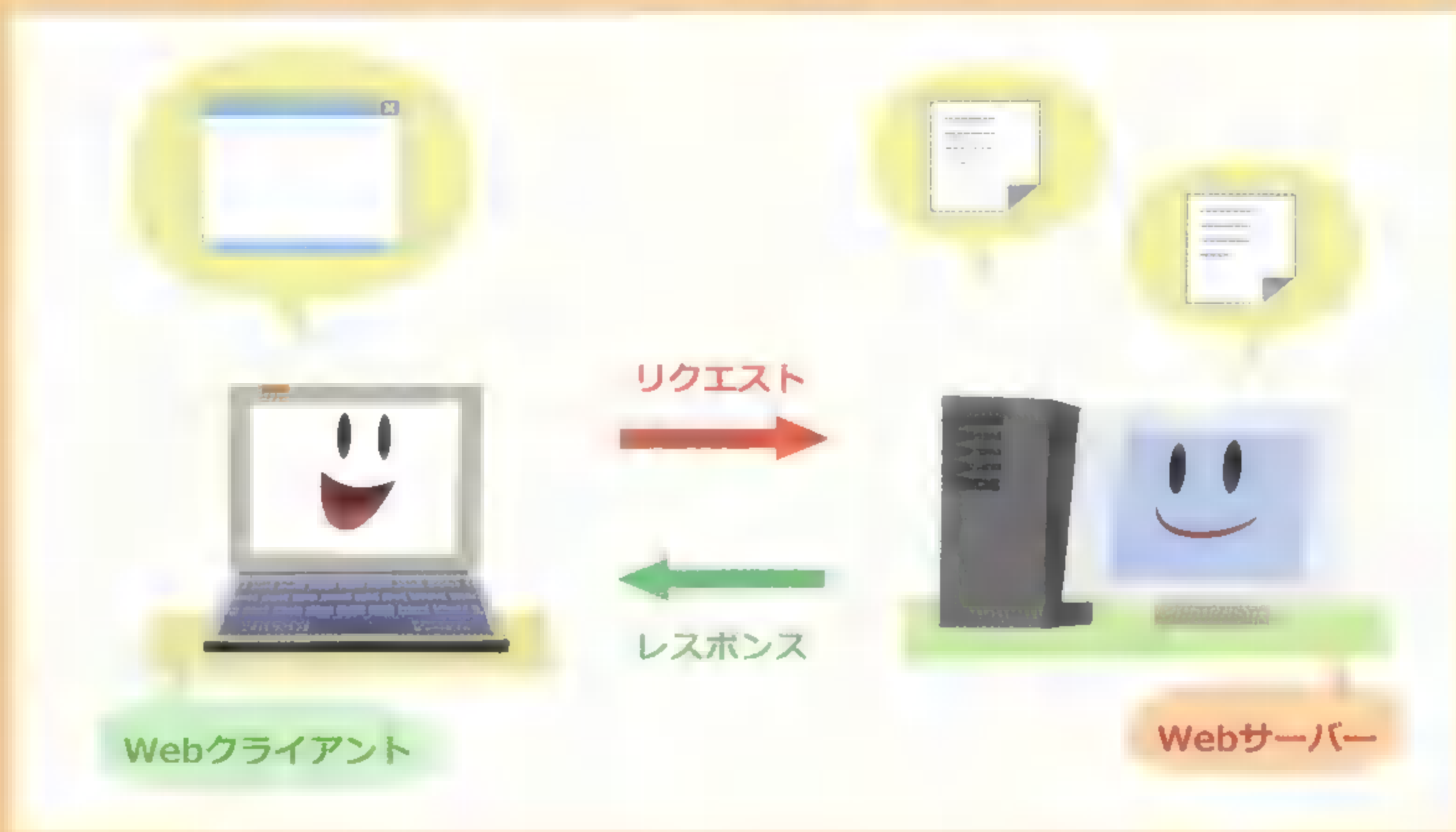
マーカーは地図上の1点をあらわすために使われる記号です。このほかにも、ポリライン（地図上の線）・ポリゴン（地図上の領域）・情報ウインドウなどの記号を利用することができます。これらの記号は「オーバーレイ」と呼ばれており、プログラム上でカスタマイズすることもできます。くわしくはGoogle社の情報を参照してください。

Ajax

Column

マップを表示する際には、Webサーバーとの通信が行われています。JavaScriptでは、Webサーバーと通信するために、Ajax（Asynchronous〔非同期〕JavaScript XML）と呼ばれる通信の仕組みを使います。Ajaxは、JavaScriptの処理中にWebサーバーとの通信を行う仕組みです。

Webサーバーに対してデータを要求することを、リクエスト（request）といいます。サーバーからの応答を、レスポンス（response）といいます。Ajaxは、このリクエスト・レスポンスによる通信を、JavaScriptのコードの通常の処理とは別に（非同期に：Asynchronous）行うことができるということを意味しています。

■非同期に通信を行うAjax

JavaScriptのプログラム中では、XMLHttpRequestオブジェクトの機能を使って、直接Webサーバーと通信することもできます。Webサーバーと連携することで、高度なWebサイトを構築していくことができるのです。

6-4 緯度・経度情報を使おう

チャレンジ!! 住所情報を表示する

次に、マウスでクリックした位置情報から住所情報を検索し、表示するページを作成してみましょう。たとえば、マップ上の「富士山」をクリックした場合に、富士山近辺の住所が表示されるといった具合です。マップを利用することで、住所情報を調べることができるようになっているのです。

■マップの位置情報を使いこなす

住所情報を検索するには？

住所情報を表示するには、クリックした位置の緯度・経度情報から住所を検索したうえで、検索結果を処理する必要があります。順を追ってみてみましょう。

緯度・経度情報から検索する

まず、住所の検索を行います。このためには、Geocoderオブジェクトを作成します（❶）。このgeocode()メソッドに、緯度・経度情報を渡して検索を行うのです。

クリックした位置を調べ（❷）、緯度・経度情報を作成します（❸）。

そして、この情報をgeocode()メソッドに指定して検索を行います（❹）。このとき、検索結果を受け取る処理も指定しておきます（❺）。

```
function map_click(e)
{
   var geo = new google.maps.Geocoder();

   cp = e.latLng;
   var ltlg = {latLng: cp};
   geo.geocode(ltlg, getResult)
}
```

❶Geocoderオブジェクトを作成します
❷クリックした位置の緯度・経度情報を調べます
❸緯度・経度情報を作成します
❹緯度・経度から検索を行います
❺処理を指定しておきます

検索結果を表示する

結果を受け取るgetResult()処理の中では、2つの変数を指定しておきます（❶）。1つ目の変数は位置情報を、2つ目の変数は検索の状態を受け取るためのものです。

そこで、2つ目の変数statusで、検索の状態を調べて検索が成功していれば（❷）、1つ目の変数resから位置情報を取得します。

```
function getResult(res, status)
{
   if(status == google.maps.GeocoderStatus.OK){
      if(res[0] != null){
      map.setCenter(cp);
```

❶検索結果を受け取ったときに処理されます
❷検索が成功した場合に・・・
❸位置情報が存在すれば・・・
❹マップの中央にして移動します

```
            var mp = {
                map: map,
                position: cp
            };
            var mk = new google.maps.Marker(mp);
            var ad = document.getElementById("address");
            ad.innerHTML = res[0].formatted_address;
            }
        }
        else{
            window.alert("利用できませんでした。");
        }
    }
```

❺マーカーを作成します

❻住所情報を表示します

検索が失敗した場合の処理です

位置情報が存在すれば（❸）、この位置にマップの中央を移動して（❹）、マーカーを作成します（❺）。

最後に、Webページ中の指定位置に住所情報を表示しています（❻）。

ジオコーディング

緯度・経度と住所情報の検索を行う機能をジオコーディングと呼んでいます。Geocoderオブジェクトには、ジオコーディング機能がまとめられています。

なお、指定した緯度・経度によっては、住所情報が存在しない場合もありますので注意してください。おおよその位置・情報を調べる場合に活用すると便利でしょう。

6-5 マップを完成させる

マップを完成させよう

それでは、住所情報を検索するマップを完成させましょう。クリックした位置の住所情報を表示します。

Sample6-5-1.html ■ マップを完成させる

```
<!DOCTYPE html>
<html lang="ja">
<head>
<link rel="stylesheet" href="Map.css">
<title>サンプル</title>
<script type="text/javascript" src="http://maps.google.com/maps/api/js?key=XXXXX&sensor=
false">
</script>
<script type="text/javascript">
var map;
var cp;

function init()
{
   var p =  new google.maps.LatLng(36,140);
   var opt = {
      zoom: 8,
      center: p,
      mapTypeId: google.maps.MapTypeId.ROADMAP
   };
   var mm = document.getElementById("mymap");
   map = new google.maps.Map(mm, opt);

  google.maps.event.addListener(map, "click", map_click);
}
function map_click(e)
{
   var geo = new google.maps.Geocoder();
```

Geocoderオブジェクトを作成します

```
    cp = e.latLng;
    var ltlg = {latLng: cp};
    geo.geocode(ltlg, getResult)
}
function getResult(res, status)
{
    if(status == google.maps.GeocoderStatus.OK){
        if(res[0] != null){
        map.setCenter(cp);
        var mp = {
            map: map,
            position: cp
        };
        var mk = new google.maps.Marker(mp);
        var ad = document.getElementById("address");
        ad.innerHTML = res[0].formatted_address;
        }
    }
    else{
        window.alert("利用できませんでした。");
    }
}
</script>
</head>
<body onload="init()">
<div id="mymap"></div>
<form name="myform">
<h3 id="address"></h3>
</form>
</body>
</html>
```

- クリックした位置の緯度・経度情報を調べます
- 緯度・経度情報を作成します
- 緯度・経度から検索を行います
- 検索結果を受け取ったときに処理されます
- 検索が成功した場合に・・・
- 位置情報が存在すれば・・・
- マップの中央にして移動します
- マーカーを作成します
- 住所情報を表示します

■ Sample6-5-1の表示

マップ上の東京湾近辺をクリックしてみました。住所情報が表示されています。

さて、いかがでしたでしょうか？ JavaScriptを使って、バリエーションに富んだWebサイト・Webアプリを作成することができたでしょうか？

JavaScriptを使えば、さらに高度なプログラムも作成していくことができるでしょう。JavaScriptを使って自由なWebサイトを構築していってみてください！

Quick Reference

■ Dateオブジェクト（日付に関する機能）

メソッド	説明
getFullYear()	年を取得する
getMonth()	月を取得する
getDate()	日を取得する
getDay()	曜日を取得する
getHours()	時を取得する
getMinutes()	分を取得する
getSeconds()	秒を取得する
getMilliseconds()	ミリ秒を取得する
setFullYear(year)	年を設定する
setMonth(month)	月を設定する
setDate(date)	日を設定する
setDay(day)	曜日を設定する
setHours(hour)	時を設定する
setMinutes(min)	分を設定する
setSeconds(sec)	秒を設定する
setMilliseconds(msec)	ミリ秒を設定する
toDateString()	日付文字列を取得する
toTimeString()	時刻文字列を取得する
toLocaleDateString()	ローカル日付文字列を取得する
toLocaleTimeString()	ローカル時刻文字列を取得する
toString()	文字列を取得する

■ Stringオブジェクト（文字列に関する機能）

メソッド	説明
indexOf(str)	前方からstrを検索する
lastindexOf(str)	後方からstrを検索する
substring(s,e)	s文字目～e文字目を取得する
substr(s, n)	s文字目からn文字取得する
split(str)	strで分割した文字列配列を取得する
toUpperCase()	大文字に変換する
toLowerCase()	小文字に変換する
charAt(n)	n文字目を取得する
link(str)	リンクを作成する
match(pt)	パターンを検索
replace(pt,str)	パターンを検索してstrで置換
search(pt)	パターンを検索して最初の文字を取得
concat(str)	strを追加する
length	文字列の長さ

■ Mathオブジェクト（数値に[illegible]する機能）

[illegible]	[illegible]
ceil(num)	切り上げて整数を取得
floor(num)	切り捨てて整数を取得
round(num)	四捨五入で整数を取得
abs(num)	絶対値を取得
max(num1, num2)	最大を取得
min(num1, num2)	最小を取得
pow(num, p)	p乗を取得
sqrt(num)	平方根を取得
sin(num)	サインを取得
cos(num)	コサインを取得
tan(num)	タンジェントを取得
random()	0以上1未満の乱数を取得
PI	π

■ Windowオブジェクト（ウィンドウに関する機能）

[illegible]	[illegible]
open(url, name, opt)	URL・ウィンドウ名・オプションを指定してウィンドウをオープンする
close()	ウィンドウをクローズする
focus()	ウィンドウを前面にする
alert(str)	メッセージを指定して警告ダイアログ表示する
confirm(str)	メッセージを指定して確認ダイアログ[illegible]示する
moveBy(x, y)	相対位置へ[illegible]
moveTo(x, y)	絶対位置へ[illegible]
resizeBy(x, y)	相対サイズ変更
resizeTo(x, y)	絶対サイズ変更
scrollBy(x, y)	相対位置へスクロール
scrollTo(x, y)	絶対位置へスクロール
print()	印刷
setInterval(func, num)	指定した間隔でタイマーの処理を行う
setTimeOut(func, num)	指定したミリ秒後にタイマーの処理を行う
clearInterval()	setInterval()の指定を解除する
clearTimeOut()	setTimeOut()の指定を解除する
width	[illegible]
height	高さ
location	アドレスバー表示／非表示
scrollbars	スクロールバー表示／非表示
resizable	サイズ変更可能／不可能
toolbar	ツールバー表示／非表示
status	ステータスバー表示／非表示
menubar	メニューバー表示／非表示

■ Navigatorオブジェクト（ブラウザに関する機能）

メソッド	説明
appName	ブラウザ名
appversion	ブラウザバージョン
platform	プラットフォーム
userAgent	ユーザーエージェント

■ Locationオブジェクト（ロケーションに関する機能）

メソッド	説明
reload()	再読み込み
replace(url)	指定ページに移動
href	現在のURLを取得／リンク先を設定
hash	ページ中のリンクを取得／リンク先を設定
host	ホストに関する情報
hostname	ホスト名
pathname	パス名
port	ポート番号
protocol	プロトコル
search	クエリ

■ Documentオブジェクト（HTML文書に関する機能）

メソッド	説明
clear()	文書内容をクリアする
open()	文書出力を開始する
close()	文書出力を終了する
write()	文書に出力する
writeln()	文書に出力・改行する
getSelection()	選択文字列を取得する
cookie	クッキーの[illegible]
domain	文書のドメイン
lastModified	文書の最終更新日
referrer	リンク元URL
title	タイトル
location	文書のURL
URL	文書のURL

■ DOM

メソッド	説明
Documentオブジェクト	
getElementByID()	idを指定して要素を取得する
getElementsByTagName()	タグ名を指定して要素リスト取得する
createElement(name)	要素名を指定して要素を作成する

[illegible]	[illegible]
createAttribute(name)	属性名を指定して属性を作成する
createTextNode(text)	テキストを指定してテキストノードを作成する
createComment(text)	テキストを指定してコメントを作成する
Nodeオブジェクト	
appendChild(elem)	指定したノードを子ノードとして追加する
removeChild(elem)	指定した子ノードを削除する
replaceChild(elem1, elem2)	指定した子ノード1を子ノード2で置き換える
innerHTML	ノード内のHTML部分
nodeType	ノード種類 [illegible]1＝要素　2＝属性　3＝テキスト　など
nodeName	ノード名
nodeValue	ノード[illegible]
childNodes	子ノードのリスト
firstChild	最初の子ノード
lastChild	最後の子ノード
previousSibling	前の兄弟ノード
nextSibling	次の兄弟ノード
Elementオブジェクト	
createAttribute(name)	指定した属性名の属性を作成する
removeAttribute(name)	指定した属性を削除する
getAttribute(name)	指定した属性名の属性値を取得する
setAttribute(name, value)	属性名と属性値を設定する
attributes	属性のリスト

■2dコンテキスト（キャンバスに関する[illegible]）

[illegible]	[illegible]
beginPath()	パスを開始する
closePath()	パスを閉じる
moveTo(x, y)	移動する
lineTo(x, y)	指定位[illegible]で[illegible]を引く
arc(x, y, r, s, e, c)	(x, y) を左上端とする半径r、角度s〜eの円弧を描く (c=true/false：左回り／右回り)
arcTo(x1, y1, x2, y2, r)	(x1, y1) と (x2, y2) を半径rの円弧でつなぐ
stroke()	[illegible]で描画する
fill()	塗りつぶしで描画する
fillRect(x, y, w, h)	(x, y) を左上端とする[illegible]w高さhの塗りつぶし四角形を描く
storkeRect(x, y, w, h)	(x, y) を左上端とする[illegible]w高さhの四角形を描く
clearRect(x, y, w, h)	(x, y) を左上端とする[illegible]w高さhの四角形を削除する
strokeText(txt, x, y)	(x, y) から線テキストを描く
fillText(txt, x, y)	(x, y) から塗りつぶしテキストを描く
clip()	クリッピングする
createLinearGradient(x0, y0, x1, y1)	(x0, y0) から (x1, y1) にグラデーション
createRadialGradient(x0, y0, r0, x1, y1, r1)	(x0, y0, r0) から (x1, y1, r1) にグラデーション
drawImage(img, x, y)	(x, y) から画像を描画する

メソッド	説明
drawImage(img, x, y, w, h)	(x, y)から幅w高さhで画像を描画する
createPattern(img, repeat)	パターンを作る
scale(x, y)	変形する
strokestyle	線色
lineWidth	線の幅
lineCap	線の端（butt/ round/square）
fillStyle	塗りつぶし色
shadowBlur	影
shadowColor	影色
shadowOffsetX	X方向の影幅
shadowOffsetY	Y方向の影高さ

リソース

HTML

- HTML 4.01
  http://www.w3.org/TR/1999/REC-html401-19991224/
- HTML5
  http://www.w3.org/TR/html5/

CSS

- CSS
  http://www.w3.org/Style/CSS/

JavaScript

- Mozilla Developers Network
  https://developer.mozilla.org/ja/docs/Web/JavaScript

Googleマップ

- Google Maps JavaScript API V3
  http://code.google.com/intl/ja/apis/maps/documentation/javascript/

JavaScriptライブラリ

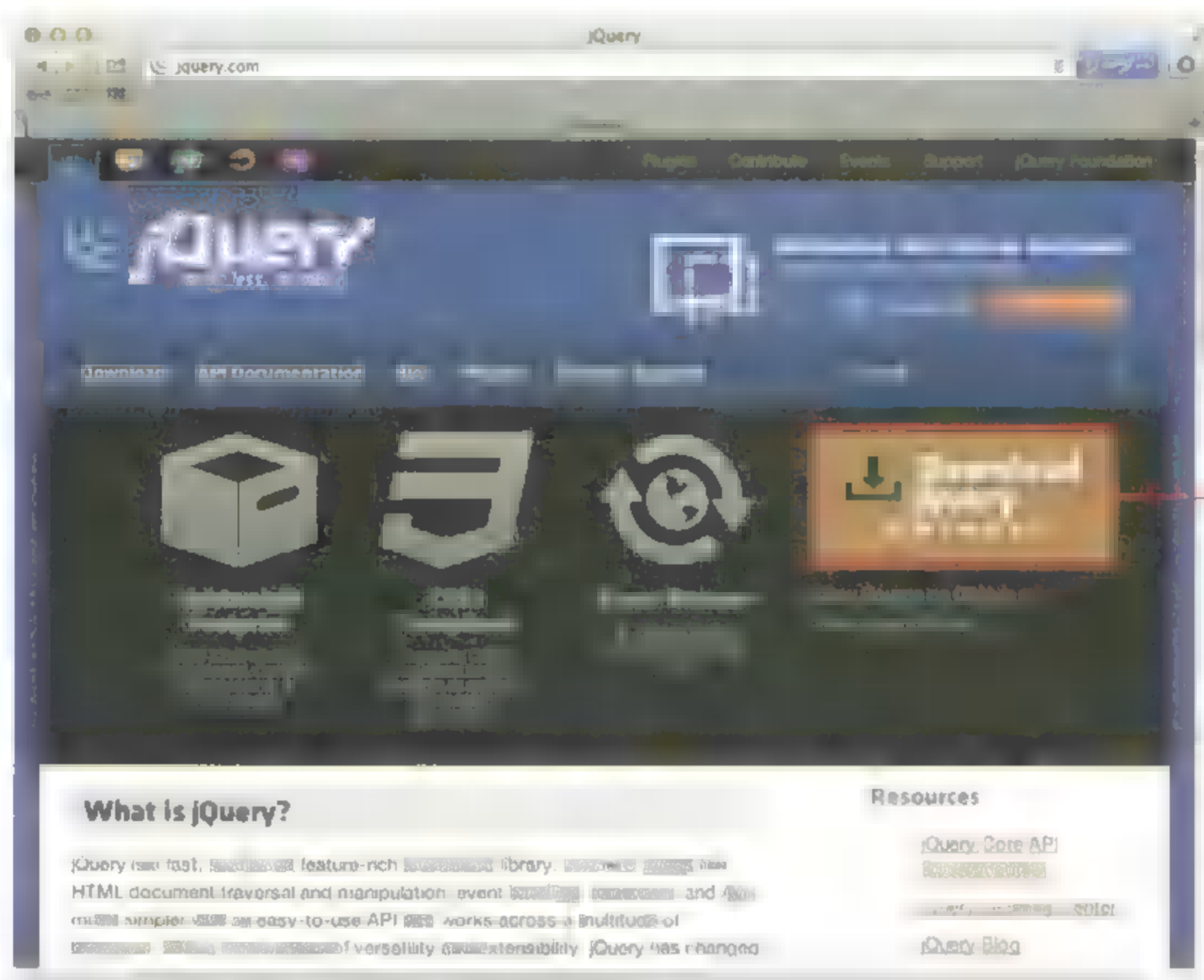
- jQuery
  http://jquery.com/

jQuery

JavaScriptのライブラリとしてjQueryが広く利用されています。jQueryもJavaScriptと同様にHTMLファイル内にプログラムを書くことで利用することができます。ここでは、かんたんにjQueryの使い方を紹介しておきましょう。

1. ライブラリをダウンロードする

jQueryのサイトからライブラリファイルをダウンロードします。

http://jquery.com/

2. ライブラリを配置する

ダウンロードしたライブラリファイル（本書では「jquery-1.10.2.min.js」）を、jQueryを使用するHTMLファイルと同じフォルダ内に保存してください。

3. HTMLファイルを作成・表示する

jQueryを利用するためのHTMLファイルを作成します。このためには「script」要素に次の指定が必要になります。この指定は、ほかの「script」要素とは別の要素として指定する必要があります。なお、ここではライブラリのファイル名として「jquery-1.10.2.min.js」を指定しています。

ダウンロードしたファイル名を指定します

```
<script type="text/javascript" src="jquery-1.10.2.min.js"></script>
```

Appendix.html ■ jQueryを使用したプログラム

```
<!DOCTYPE html>
<html lang="ja">
<head>
<link rel="stylesheet" href="Sample.css">
<title>サンプル</title>
<script type="text/javascript" src="jquery-1.10.2.min.js"></script>
<script type="text/javascript">
function welcome()
{
  $("#image").fadeTo(300, 0.5);
}
</script>
</head>
<body>
<img src="Sample.jpg" id="image" onclick= "welcome()"/>
</body>
</html>
```

ライブラリを指定します

画像を・・・

フェード処理します

■ Appendix.htmlの表示

画像をクリックすると、徐々に画像がフェードアウトします。jQueryを使うと、このように見栄えのよいWebや高度な機能をかんたんに実現することができます。

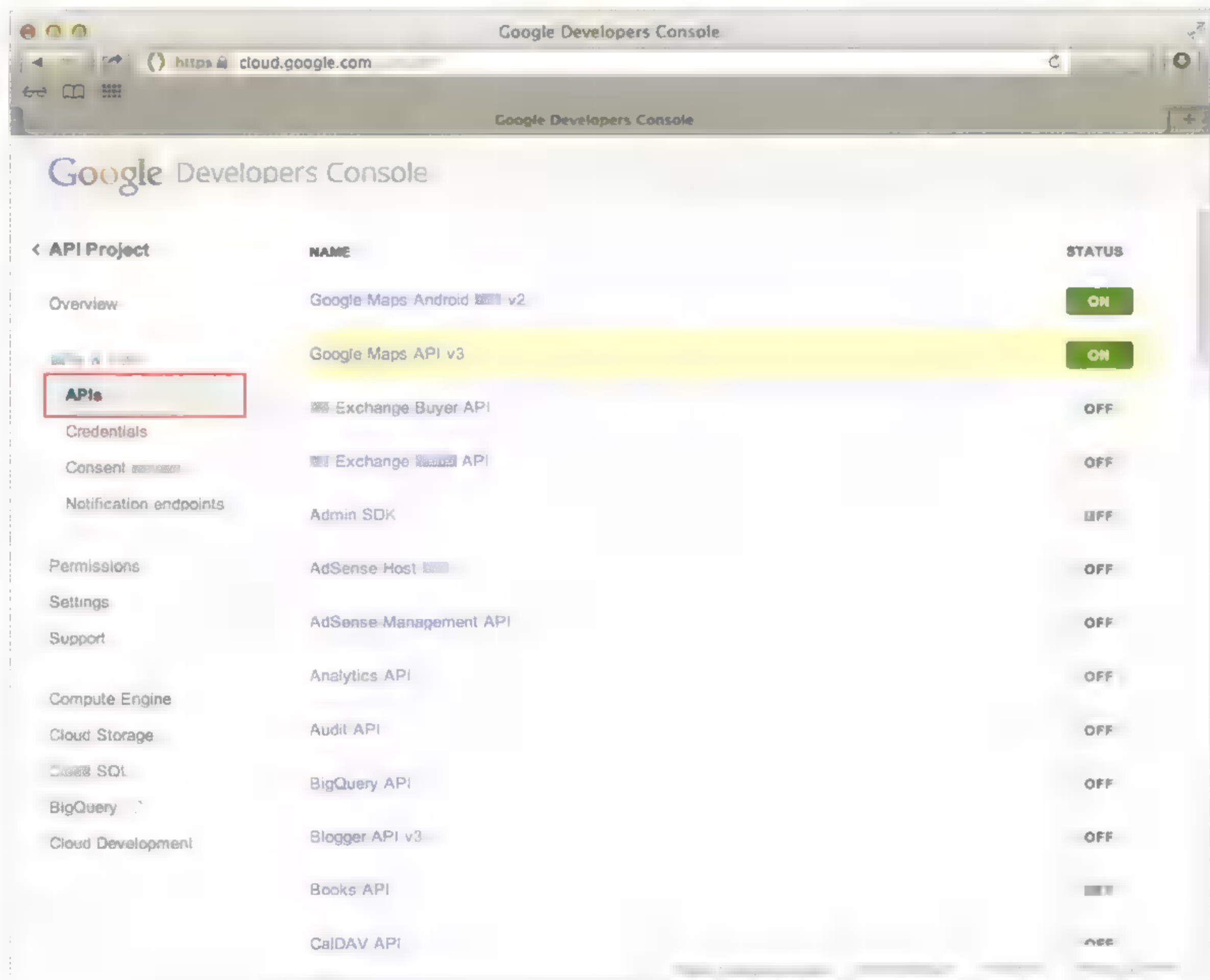
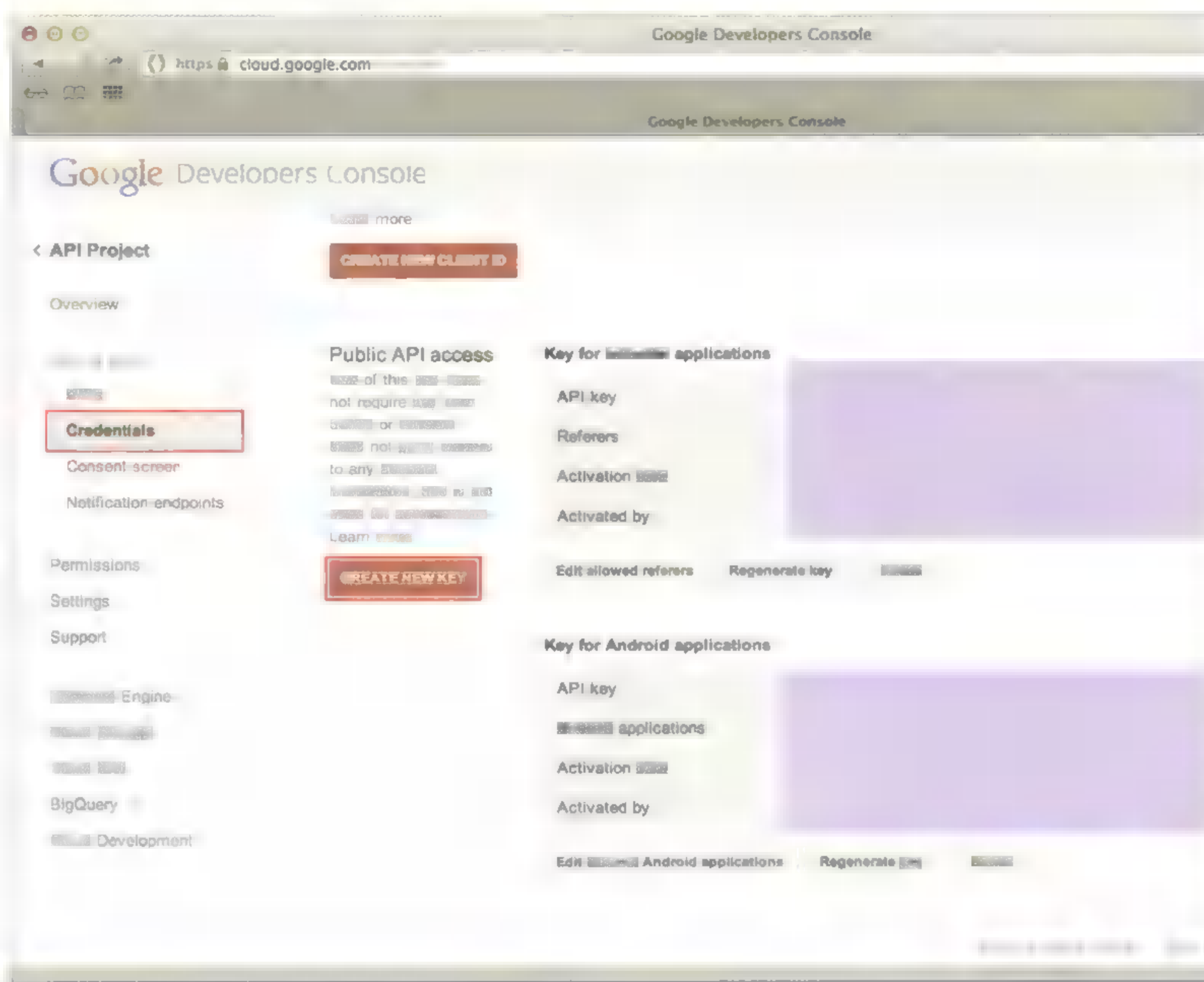
Google Maps APIキー

Googleマップ（第6章）の利用にあたってはAPIキーの取得が推奨されています。ここではAPIキーの取得・作成方法を説明します。

1. 以下の「Google APIs Console」ページにサインインします。サインインにはGoogleアカウントが必要です。ページ上の「Create project...」ボタンをクリックします。

   Google APIs Console

   URL https://code.google.com/apis/console/

2. 「APIs & auth」→「APIs」にアクセスして「Google Maps API v3」を選択します。ライセンス同意書に同意して「Accept」ボタンを押します。

3. 「Credentials」にアクセスして、「CREATE NEW KEY」からAPIキーを取得します。

索引

記号

" " 34
//（コメント） 33
+ 35
;（セミコロン） 24
<（未満） 42
<=（以下） 42
==（等価） 42
>（より大きい） 42
>=（以上） 42
&&（論理積） 43
!（論理否定） 43
!=（非等価） 42
||（論理和） 43

数字

1行の表を作成する 50
2d 151
2dコンテキスト 197

英字

Ajax 185
appendChild()メソッド 97
canvas要素 151
case 41
clearInterval()メソッド 108
createTextNode()メソッド 97
CSS 18, 198
CSSファイル 142
Dateオブジェクト 30, 194
　getDay()メソッド 36
Documentオブジェクト 35, 196
　getElementById()メソッド 94
　writeln()メソッド 34
DOM 92, 196
　～の主なメンバ 95
DOMオブジェクト
　setAttribute()メソッド 94
　～を特定する 94
drawImage()メソッド 158
Elementオブジェクト 197
Eventオブジェクト 169
false 168
for文 47, 48
form要素 116
function 73
getContext()メソッド 151
getDate()メソッド 32
getDay()メソッド 36
getElementById()メソッド 94
getFullYear()メソッド 31
getHours()メソッド 32
getMinutes()メソッド 32
getMonth()メソッド 32
globalAlphaプロパティ 158
Googleマップ 175, 198
Google APIs Console 201
Google Maps APIキー 201
Google Maps JavaScript API V3 .... 175, 198
HTML 15, 198
　script要素 17
　～の要素 16
HTML 4.01 198
HTML5 .... 151, 198
ID名 93
id="ID名" 93
if文 39
if～else文 38, 39
if～else if～else文 39
img要素のname属性 80
innerHTMLプロパティ 96
input要素のonclick="処理名" 77
JavaScript 2, 198
　～の入力[illegible]所 17
JavaScriptプログラム 8
JavaScriptライブラリ 198
jQuery 147, 198, 199
Locationオブジェクト 196
Mathオブジェクト 86, 195
　random()メソッド 86
name属性 116
Navigatorオブジェクト 146, 196

userAgentプロパティ 145
new 26
Nodeオブジェクト 197
null 124
onclick="処理名()" 74
ondblclick="処理名()" 76
open()メソッド 76
PC 6
random()メソッド 86
RegExpオブジェクト 123
removeChild()メソッド 97
rgb()関数 156
script要素 17
setAttribute()メソッド 94
setInterval()メソッド 103
Stringオブジェクト 194
switch文 40, 41
table要素 50
td要素 50
tr要素 50
true 168
URL 19
userAgentプロパティ 145
Webのしくみ 12
Webブラウザ 7
Webページ 2
while文 48
Windowオブジェクト 195
setInterval()メソッド 103
writeln()メソッド 34

あ行

値を記憶する 34
アニメーション 160
～を作成する 161
緯度・経度情報 186
～から検索する 187
イベント 75
～時に動作させる 75
インクリメント 48
インターネット 12
エディタ 6
演算子 55, 56
円をランダムに描画する 154
オブジェクト 24
～の種類 28
～を作成する 26
～を操作する 94
～を利用する 27

か行

隠し属性 115
画像の透明度 158
画像を入れ替える 79
画像を描画する 156
カレンダー 66
関数 78
キャンバス 150, 197
～に描画する 151
～を指定する 151
クライアント 13
繰り返し処理 47, 48
繰り返しの中の繰り返し 53
繰り返し文 52
クリックしたときに処理を行う 73
計算をする 55
コード 22
コメント 33
コンテキスト 151

さ行

サーバー 13
サムネイルから選択する 99
ジオコーディング 188
時刻 22
～を表示する 32
写真のリストを作成する 49
住所情報を検索する 187
小計を計算する 134
条件 42
スタイルシート 18
スライドショー 106
正規表現 128
説明を表示する 97
セレクタ 18
選択フォームを設計する 129
選択をクリアする 135

た行

タイマー 102
～で画像を入れ替える 104
～を初期化する 103
タグ 15
ダブルクリック時に処理させる 76
端末別の表示 139
チェックボックス 115
地図 174
月・日を表示する 32
データベース 138

テキストエディタ 6
テキストエディット 7
テキストエリア 115
テキストボックス 115
テキストを入れ替える 96
デクリメント 48
年を表示する 31
ドロップダウンメニュー 115

な行

入力されたデータを調べる 117
入力をチェックする 120
ノード 93

は行

場合分け 38, 39
配列 61
パソコン 6
パターン 128
引数 101
表 52
描画に使う主なメンバ 152
描画のためのメンバ 159
ファイル 138
　～の場所を指定する 19
フォーム 114
　～の送信 119
　～のデータを調べる 117
フォーム部品 114
フォーム要素を作成する 116
ブラウザ 7
ブラウザのチェックをする 145
プログラミング言語 3, 4
プログラム 3
プロパティ 18, 29
文 24
ページの要素を操作する 91
変数 25
ボタンのクリックで表示する 77

ま行

マーカー 181
マーカーを表示する 182
マウスオーバー 97
マウスでペイントする 166
マップ 174
　～の種類 180
　～の準備方法 176
　～を完成させる 189
　～を表示する 178
見出しテキストをランダムに表示する 89
メールアドレス 125
メソッド 29
メモ帳 7
メンバ 29
文字列 34

や行

ユーザーの操作で動作させる 72
郵便番号 123
要素 15
曜日名を表示する 64

ら行

ライブラリ 146
ラジオボタン 115
ランダムに入れ替える 87
ランダムに画像を表示する 84
リクエスト 185
リスト 52
リストボックスメニュー 115
レスポンス 185
連想配列 133
論理積 43
論理否定 43
論理和 43

●著者略歴
高橋 麻奈
1971年東京生まれ。東京大学経済学部卒業。主な著作に『やさしいC』『やさしいC++』『やさしいC#』『やさしいC アルゴリズム編』『やさしいJava』『やさしいJava 活用編』『やさしいJava オブジェクト指向編』『やさしいXML』『やさしいPHP』『やさしいAndroidプログラミング』『やさしいWebアプリプログラミング』『やさしいiOSプログラミング』『プログラムのからくりを解く』『やさしいITパスポート講座』『やさしい基本情報技術者講座』『やさしい応用情報技術者講座』『やさしい情報セキュリティスペシャリスト講座』(SBクリエイティブ)、『入門テクニカルライティング』『ここからはじめる統計学の教科書』(朝倉書店)、『心くばりの文章術』(文藝春秋)、『マンガで学ぶ 医療統計』(みみずく舎/医学評論社) などがある。

●本書のサポートページ (サンプルダウンロード)
http://homepage3.nifty.com/~mana/yasajs.html

やさしいJavaScript(ジャバスクリプト)のきほん

2014年 2月 7日　　初版第1刷発行

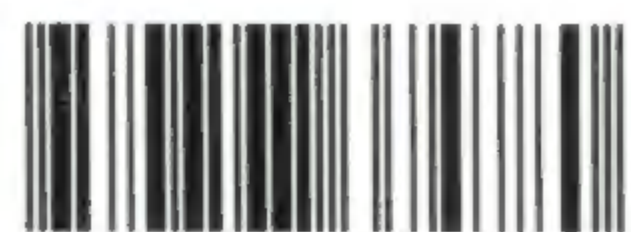
著　者　高橋 麻奈(たかはし まな)
制　作　風工舎
発行者　小川 淳
発行所　SBクリエイティブ株式会社
〒106-0032 東京都港区六本木2-4-5
営業 03-5549-1201
印　刷　株式会社シナノ
装丁デザイン・本文イラスト　YOUCHAN (トゴル・カンパニー)

落丁本、乱丁本は小社営業部にてお取り替えします。
定価はカバーに記載されています。

Printed in Japan　　ISBN978-4-7973-7478-0

SB Creative

ISBN978-4-7973-7478-0
C0055 ¥2000E

定価 本体2,000円 +税

やさしいJavaScriptのきほん

高橋麻奈 著

SB Creative